USO DE REDES NEURAIS CONVOLUCIONAIS PARA DETECÇÃO DE CÂNCER EM MAMOGRAFIAS

BRENO BORRO MACEDO

TRABALHO DE CONCLUSÃO DE CURSO
EM ENGENHARIA ELÉTRICA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

Uso de Redes Neurais Convolucionais para detecção de câncer em mamografias

Breno Borro Macedo

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO(A) ELETRICISTA.

APROVADA POR:

Eduardo Peixoto Fernandes da Silva, Ph.D. (ENE/UnB)
(Orientador)

Alexandre Ricardo Soares Romariz, Ph.D. (ENE/UnB))
(Examinador Interno)

João Luiz Azevedo de Carvalho, Ph.D. (ENE/UnB)
(Examinador Interno)

Brasília/DF, Julho de 2024.

FICHA CATALOGRÁFICA

BORRO MACEDO, BRENO

Uso de Redes Neurais Convolucionais para Detecção de Câncer em Mamografias. [Brasília/DF] 2024.

210 x 297 mm (ENE/FT/UnB, Engenheiro(a) Eletricista, Trabalho de Conclusão de Curso, 2024).

Universidade de Brasília, Faculdade de Tecnologia, Departamento de Engenharia Elétrica.

Departamento de Engenharia Elétrica

1. Breast cancer
2. Tomosynthesis
3. Convolutional Neural Networks
4. Transfer Learning
5. DenseNet-121
6. ResNet-50
7. VGG-16
8. Python

I. ENE/FT/UnB

II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

BORRO MACEDO, BRENO (2024). Uso de Redes Neurais Convolucionais para Detecção de Câncer em Mamografias. Trabalho de Conclusão de Curso, Publicação ENE 07/2024, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF.

CESSÃO DE DIREITOS

AUTOR: Breno Borro Macedo

TÍTULO: Uso de Redes Neurais Convolucionais para Detecção de Câncer em Mamografias.

GRAU: Engenheiro(a) Eletricista ANO: 2024

Breno Borro Macedo

Universidade de Brasília (UnB)

Campus Darcy Ribeiro

Faculdade de Tecnologia - FT

Departamento de Engenharia Elétrica (ENE)

Brasília - DF CEP 70919-970

AGRADECIMENTOS

Agradeço em primeiro lugar à minha família e minha namorada, que sempre me deram ouvidos e estiveram ao meu lado em tempos difíceis, sempre me motivando a fazer aquilo que gosto e que motiva, me incentivando a buscar a felicidade nos lugares certos.

Deixo também meus agradecimentos aos vários amigos que fiz durante meu período na graduação e aqueles que há tempo conheço, que foram essenciais em me proporcionar bons momentos e memórias para guardar comigo.

Agradeço também ao meu orientador, Eduardo Peixoto, por orientar meus trabalhos e estudos durante o período de formulação deste trabalho.

Por fim, mas não menos importante, agradeço à comunidade de forró da Universidade de Brasília por me proporcionar um espaço de aconchego e conforto, o qual me ajudou a crescer muito como pessoa e trabalhar em mim mesmo.

RESUMO

O câncer de mama é uma das principais causas de mortalidade entre mulheres no mundo e representa um desafio para a saúde pública. Sua detecção precoce é de grande importância, pois pode auxiliar no sucesso do tratamento e na sobrevivência das pacientes. Os diagnósitcos dessa doença são realizados por um médico radiologista, que analisa a imagem de uma mamografia gerada por técnicas como tomossíntese ou radiografia e destaca o local do tumor, caso exista. Como tradicionalmente o processo de diagnostóstico é realizado com imagens, isso possibilita que sejam utilizadas técnicas de processamento de imagens ou o desenvolvimento de redes neurais artificiais que sejam capazes de realizar esse papel de detecção de câncer.

Este trabalho explora o potencial das Redes Neurais Convolucionais (CNNs) na detecção de câncer de mama ao utilizar técnicas de *transfer learning* com as arquiteturas DenseNet-121, ResNet-50 e VGG-16. As redes foram treinadas para identificar características específicas do câncer de mama a partir do uso da linguagem de programação *python* e a biblioteca *tensorflow* para o processamento e organização das imagens do *dataset* público *DukeDBT* e também o desenvolvimento das redes neurais.

O objetivo do projeto é avaliar a acurácia, especificidade e sensibilidade das CNNs na detecção de câncer de mama e destacar a viabilidade de sua aplicação em ambientes clínicos com recursos limitados, oferecendo uma alternativa para melhorar os diagnósticos e possivelmente reduzir a mortalidade associada à doença.

Os resultados obtidos no trabalho mostram uma viabilidade no uso dessas técnicas para a detecção de tumores. Como exemplo, a rede VGG-16 conseguiu classificar corretamente as imagens que possuíam efetivamente tumores com 83% de sucesso e, possivelmente, mais trabalhos e pesquisas nessa área poderiam melhorar a qualidade dessas classificações.

Palavras-chave: Câncer de mama, Tomossíntese, Redes Neurais Convolucionais, *transfer learning*, DenseNet-121, ResNet-50, VGG-16, *python*, *tensorflow*

ABSTRACT

Breast cancer is one of the leading causes of mortality among women worldwide and represents a significant public health challenge. Early detection is crucial, since it can improve treatment success and patient survival. Diagnosing this disease is typically performed by a radiologist who analyzes mammography images, generated through techniques like tomosynthesis or radiography, to highlight the tumor location, if present. Since the diagnostic process traditionally relies on images, it opens up the possibility of employing image processing techniques or developing artificial neural networks capable of performing cancer detection.

This study explores the potential of Convolutional Neural Networks (CNNs) in breast cancer detection using transfer learning techniques with DenseNet-121, ResNet-50, and VGG-16 architectures. The networks were trained to identify specific breast cancer characteristics using the Python programming language and the TensorFlow library for processing and organizing images from the public DukeDBT dataset and for developing the neural networks.

The project's objective is to evaluate the accuracy, specificity, and sensitivity of CNNs in detecting breast cancer and to highlight the viability of their application in resource-limited clinical settings, providing an alternative to improve diagnoses and potentially reduce mortality associated with the disease.

The results obtained indicate the viability of using these techniques for tumor detection. For instance, the VGG-16 network successfully classified images containing tumors with an 83% success rate and further research and development in this area could possibly enhance the quality of these classifications.

Keywords: Breast cancer, Tomosynthesis, Convolutional Neural Networks, Transfer Learning, DenseNet-121, ResNet-50, VGG-16, Python, Tensorflow

SUMÁRIO

LISTA DE FIGURAS

CAPÍTULO 1

INTRODUÇÃO

O câncer de mama é uma das principais causas de mortalidade entre mulheres em todo o mundo e representa um dos maiores desafios de saúde pública global. Sua detecção precoce é fundamental para aumentar as chances de tratamento bem-sucedido e a sobrevivência das pacientes. Apesar disso, a escassez de radiologistas em muitos países, especialmente nos em desenvolvimento, compromete a eficácia dos diagnósticos e agrava a situação das pessoas que sofrem com essa doença (DOC, 2022) (LANES, 2022) (XU, 2023).

No Brasil, por exemplo, a distribuição desigual de radiologistas é um problema significativo. Regiões como o Norte e o Nordeste enfrentam um déficit desses profissionais quando comparadas às outras, o que resulta em longos tempos de espera para a realização e análise de mamografias. Esse cenário leva a diagnósticos possivelmente atrasados, o que aumenta o risco de mortalidade entre as pacientes. A falta de radiologistas não é um problema exclusivo do Brasil, pois até mesmo em países como os Estados Unidos a carência desses profissionais também afeta a qualidade do atendimento médico e a eficácia dos diagnósticos. A Figura 1.1 mostra uma distribuição de radiologistas por milhão de pacientes globamente (DOC, 2022) (LANES, 2022) (COLANGELO, 2022).

Nesse contexto, as inteligências artificiais, em especial as Redes Neurais Convolucionais (CNNs), podem ser utilizadas como uma solução para auxiliar no diagnóstico de câncer de mama. As CNNs são uma classe de redes neurais profundas que se destacam no processamento e análise de imagens, como mamografias. Essas redes podem ser treinadas para reconhecer padrões e anomalias com alta precisão (COLANGELO, 2022) (NG, 2024).

A aplicação de CNNs na radiologia pode ser muito vantajoso, visto que elas podem processar grandes volumes de dados rapidamente, o que permite a análise eficiente de mamografias e a detecção precoce de tumores. Em regiões com escassez de radiologistas, as CNNs podem ser implementadas em sistemas de atendimento à distância (telerradiologia), o que beneficiaria

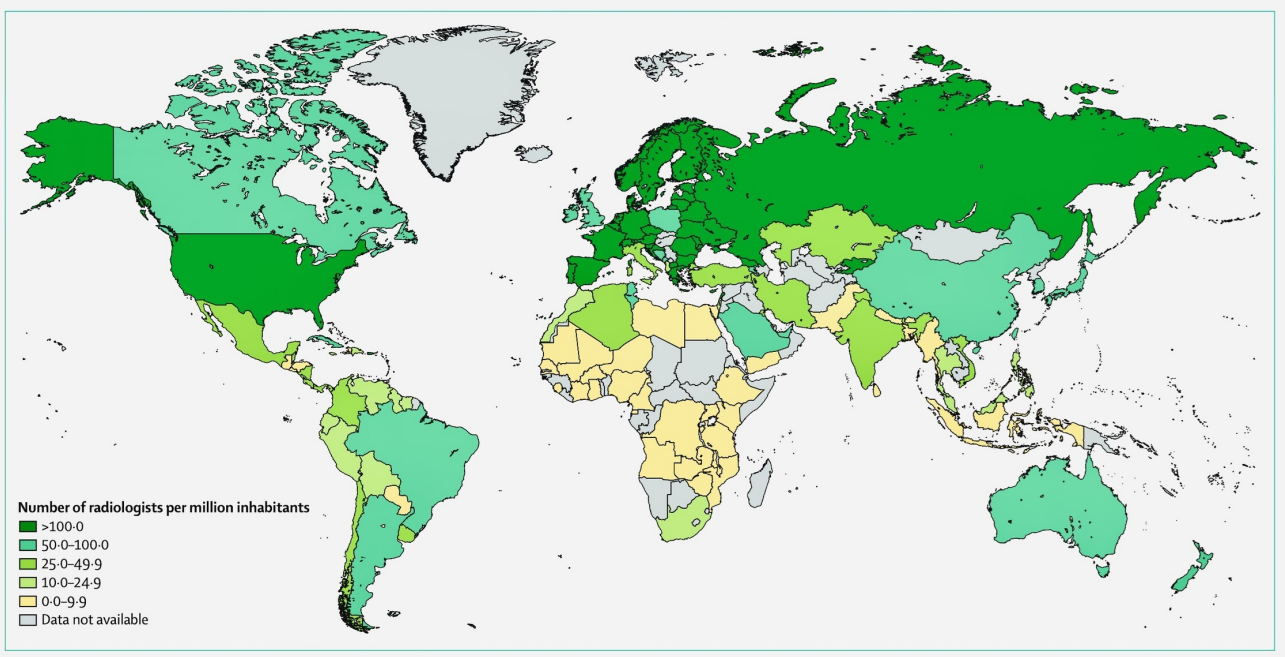

Figura 1.1. Número de ragiologistas por milhão de pacientes. É perceptível a diferença substancial do número de profissionais treinados entre os diferentes países do mundo, principalmente entre o hemisfério Norte e o Sul (COLANGELO, 2022).

áreas remotas e com poucos especialistas. Além disso, a utilização dessas tecnologias poderia aliviar a carga de trabalho dos profissionais, o que permitiria que focassem em casos mais complexos e em outras tarefas clínicas essenciais (COLANGELO, 2022) (XU, 2023). Por esses motivos, este trabalho explora o potencial de CNNs na detecção de câncer de mama e busca testar sua eficácia nessa tarefa em imagens médicas obtidas via tomossíntese.

Para o desenvolvimento deste trabalho, utilizou-se principalmente técnicas de *transfer learning*, que consiste em utilizar redes neurais pré-treinadas para outras tarefas de interesse. As arquiteturas utilizdadas incluem a DenseNet-121, ResNet-50 e VGG-16 (HUANG, 2016) (HE, 2015) (SIMONYAN, 2014).

As redes mencionadas foram ajustadas e treinadas para identificar características específicas associadas ao câncer de mama. A implementação foi realizada utilizando a linguagem de programação *python* e a biblioteca *tensorflow* para processar e organizar as imagens do *dataset* público *DukeDBT* (BUDA, 2024) e também desenvolver e treinar as redes neurais.

Este projeto busca não só avaliar a acurácia das CNNs na detecção de câncer de mama, mas também destacar a viabilidade de sua aplicação prática em ambientes clínicos com recursos limitados.

Os resultados obtidos no trabalho mostram que existe uma viabilidade no uso dessas redes neurais, pois a rede VGG-16, com uso da técnica de perda ponderada, conseguiu classificar corretamente 83% das imagens com tumores e teve uma acurácia de 90%.

CAPÍTULO 2

IMAGENS MÉDICAS VIA TOMOSSÍNTESE

A tecnologia de imagens médicas é muito utilizada na área da saúde, visto que auxilia no processo de obtenção de diagnósticos de pacientes. Entre as diversas modalidades de imagens, os arquivos de *Digital Imaging and Communications in Medicine* (DICOM) e as imagens de seios obtidas via tomossíntese têm se destacado como ferramentas essenciais no diagnóstico de doenças complexas, como o câncer de mama (AGUILLAR, 2018).

Os arquivos DICOM são o padrão internacional para armazenamento, transmissão e visualização de imagens médicas. Esses arquivos contêm não apenas as imagens, mas também outros tipos de informações importantes para os médicos, como dados do paciente, parâmetros de aquisição e anotações clínicas (VARMA, 2012).

A tomossíntese mamária, por sua vez, representa uma evolução significativa na mamografia tradicional. Também conhecida como mamografia 3D, essa técnica avançada captura múltiplas imagens dos seios em diferentes ângulos, permitindo a reconstrução de uma imagem tridimensional da mama. Isso possibilita uma melhor visualização das estruturas internas, aumentando a sensibilidade e especificidade na detecção de lesões malignas (HELVIE, 2011).

A partir disso, neste capítulo serão explorados o funcionamento da obtenção de imagens a partir da tomossíntese mamária e a arquitetura de arquivos DICOM.

2.1 MAMOGRAFIA DIGITAL POR TOMOSSÍNTESE

As imagens contidas no *dataset* utilizado neste trabalho são imagens médicas obtidas por meio de tomossíntese. Por esse motivo, convém uma explicação do procedimento utilizado para sua obtenção.

2.1.1 Definição de tomossíntese

A mamografia digital da mama por meio de tomossíntese (DBT) é uma tecnologia em desenvolvimento para melhorar a detecção e caracterização de lesões na mama, especialmente em mulheres com mamas não gordurosas (HELVIE, 2011).

Nesta técnica, múltiplas imagens de projeção são reconstruídas permitindo a revisão visual de seções finas da mama, oferecendo o potencial para revelar cânceres ocultos por tecido normal localizado acima e abaixo da lesão. A DBT envolve a aquisição de múltiplas exposições de projeção por um detector digital a partir de uma fonte de raios-X mamográfica que se move sobre um ângulo de arco limitado. Esses conjuntos de dados de imagem de projeção são reconstruídos usando algoritmos específicos (HELVIE, 2011).

O leitor clínico é apresentado com uma série de imagens (fatias) através de toda a mama, que são lidas em uma estação de trabalho. Como cada fatia reconstruída pode ter apenas 0,5 mm de espessura, massas e margens de massa que de outra forma podem estar sobrepostas com estruturas fora do plano devem ser mais visíveis na fatia reconstruída. Isso deve permitir a visualização (detecção) e melhor caracterização de lesões não calcificadas em particular (HELVIE, 2011).

2.1.2 Técnica utilizada

O surgimento da mamografia digital e dos algoritmos de reconstrução por computador permitiu o desenvolvimento de tecnologias derivadas, incluindo a tomossíntese. Na mamografia digital convencional, uma mama comprimida é exposta à radiação ionizante. A energia que passa pela mama é transformada em um sinal elétrico por um detector que produz a imagem clínica. O tubo de raios-X é estacionário, a mama é estacionária e o detector é estacionário (HELVIE, 2011).

A imagem produzida em uma única projeção (vista) é uma representação bidimensional do espaço tridimensional. Cada pixel é, portanto, uma média das informações obtidas através da espessura total da mama. Uma representação tridimensional da mama seria vantajosa, semelhante às representações tridimensionais permitidas pela tomografia computadorizada (TC),

ressonância magnética (RM) ou ultrassonografia (HELVIE, 2011).

Na tomossíntese digital da mama, o tubo de raios-X é movido através de um ângulo de arco limitado enquanto a mama é comprimida e uma série de exposições são obtidas . Essas exposições individuais representam apenas uma fração da dose total de radiação usada durante a mamografia digital convencional. Se houver um arco de movimento de 45 graus e uma exposição for feita a cada 3 graus, haverá 15 exposições individuais . Esses conjuntos de dados brutos de projeção requerem reconstrução usando algoritmos semelhantes aos usados em outros conjuntos de imagens tridimensionais (HELVIE, 2011).

Os conjuntos de dados de projeção geralmente não são interpretados pelos radiologistas, mas sim a interpretação é baseada apenas nas imagens de tomossíntese reconstruídas. Tipicamente, os conjuntos de dados de projeção são reconstruídos em fatias muito finas (por exemplo, 1 mm) para revisão pelo radiologista (HELVIE, 2011).

Na Figura 2.1 é possível observar uma representação de como o procedimento de obtenção de imagens de mama em diferentes ângulos é realizada em laboratório.

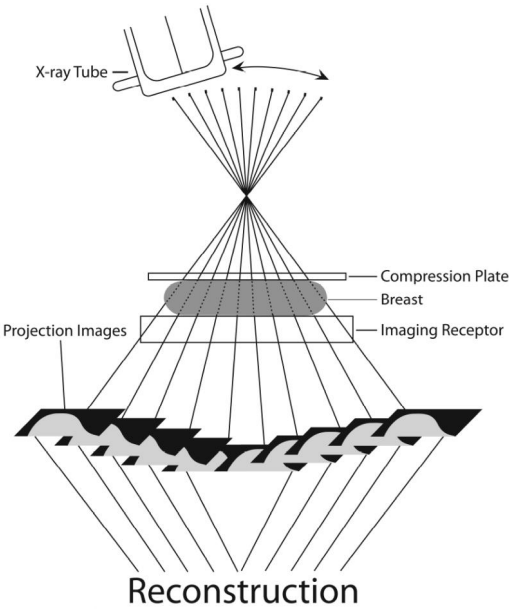

Figura 2.1. Visão esquemática da tomossíntese digital da mama. O tubo de raios-X se move através de um arco estreito enquanto a mama está comprimida. Uma série de exposições resulta em múltiplos conjuntos de dados de imagens de projeção. Os conjuntos de dados de imagens de projeção são reconstruídos em múltiplas imagens de fatias finas (por exemplo, 1 mm de espessura) para interpretação pelo radiologista (HELVIE, 2011).

É comum que diferentes posições da mama sejam obtidas por meio da tomossíntese para a análise de diagnóstico em ângulos diferentes. Como exemplo, o *dataset* utilizado para este trabalho, o *DukeDBT*, é composto de imagens de pacientes com até quatro ângulos distintos

- vista oblíqua médio-lateral esquerda (*left mediolateral oblique view* - LMLO), vista oblíqua médio-lateral direita (*right mediolateral oblique view* - RMLO), vista craniocaudal esquerda (*left craniocaudal view* - LCC), e vista craniocaudal direita (*right craniocaudal view* - RCC). Essas diferentes *views* são mostradas na Figura 2.2, para um mesmo paciente (BUDA, 2024).

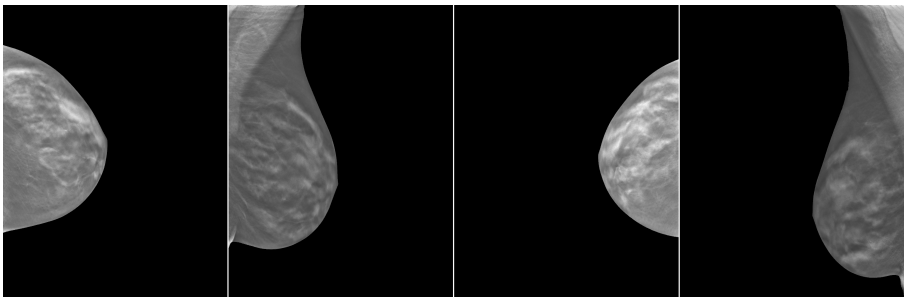

Figura 2.2. Os quatro ângulos presentes no *DukeDBT*, da esquerda para a direita, LCC, LMLO, RCC e RMLO. Todas elas são do mesmo paciente com diagnístico normal, ou seja, sem presença de tumor (BUDA, 2024).

2.1.3 Técnica de imagem

Vários fabricantes aplicaram diferentes métodos para desenvolver e realizar a tomossíntese e cada técnica possui suas vantagens e desvantagens. No entanto, essas diferenças podem produzir resultados clínicos diferentes, tornando as comparações clínicas entre os fabricantes difíceis. As restrições de engenharia incluem a dose total de radiação, tempo de imagem, movimento do paciente, desempenho do detector, movimento do detector e capacidade de imaginar toda a mama. Também é necessário fornecer capacidade futura de biópsia para aquelas lesões detectadas apenas pela tomossíntese (HELVIE, 2011).

Os fabricantes variam o arco de movimento (tipicamente de 11° a 60°), o número de exposições individuais (tipicamente entre 9 e 25) , o uso de exposição contínua ou pulsada, estabilidade ou movimento do detector, parâmetros de exposição, dose total, tamanho efetivo dos pixels, fonte de raios-X/filtro, pixels únicos ou agrupados e posição do paciente. Essas decisões teóricas e de engenharia podem levar a resultados clínicos diferentes e recomendações de leitura diferentes para os diferentes fabricantes. De particular importância é a avaliação de microcalcificações e se se tenta representar com precisão microcalcificações por DBT. Devido ao ângulo limitado de varredura, as imagens são apenas quase 3D (HELVIE, 2011).

O conjunto de dados pode ser reconstruído para o radiologista ler exibindo diferentes es-

pessuras. Por exemplo, se uma mama comprimida de 60 cm for reconstruída com espessura de 1 mm, haverá 60 fatias para o médico revisar. Se as imagens forem reconstruídas com espessuras de 0,5 mm, haverá 120 imagens a serem revisadas. Se as imagens forem reconstruídas com lâminas de 10 mm de espessura usando projeção de intensidade máxima (MIP), haverá 6 imagens a serem revisadas (HELVIE, 2011).

2.2 ARQUIVOS DICOM (.DCM)

O *dataset* utilizado neste trabalho providencia as imagens em formato DICOM (.dcm), então é de importância uma breve análise de como este tipo de arquivo é estruturado.

2.2.1 Definição de um arquivo DICOM

Todas as modalidades na prática radiológica se tornaram digitais e, portanto, lidam com imagens DICOM. Arquivos de imagem que estão em conformidade com o padrão DICOM são representados como .dcm (VARMA, 2012).

O DICOM difere de outros formatos de imagem por agrupar informações em conjuntos de dados. Um arquivo DICOM consiste em um cabeçalho e conjuntos de dados de imagem compactados em um único arquivo. As informações dentro do cabeçalho são organizadas como uma série constante e padronizada de *tags*. Ao extrair dados dessas *tags*, é possível acessar informações importantes sobre as características demográficas do paciente, parâmetros do estudo, etc (VARMA, 2012).

No interesse da confidencialidade do paciente, todas as informações que podem ser usadas para identificá-lo são removidas antes que os arquivos DICOM sejam transmitidas pela rede para fins educacionais ou outros (VARMA, 2012).

2.2.2 Estrutura de arquivos DICOM

Um arquivo DICOM consiste em um cabeçalho e conjuntos de dados de imagem, todos compactados em um único arquivo.

Os primeiros pacotes de informação em um arquivo de imagem DICOM constituem o cabeçalho. Ele armazena informações demográficas sobre o paciente, parâmetros de aquisição para o estudo de imagem, dimensões da imagem, tamanho da matriz, espaço de cores e uma série de informações não relacionadas à intensidade necessárias para que o computador exiba corretamente a imagem (VARMA, 2012).

O cabeçalho é seguido por um único atributo que contém todos os dados de intensidade de pixel da imagem. Esses dados são armazenados de forma binária de ponto fixo, com número de bits dado pelo cabeçalho, que podem ser reconstruídos como a imagem usando as suas informações (VARMA, 2012).

As informações dos dados do cabeçalho são codificadas dentro do arquivo DICOM de forma que não possam ser separadas acidentalmente dos dados da imagem. Se o cabeçalho for separado dos dados da imagem, o computador não saberá qual estudo de imagem foi realizado ou a quem pertence, e não será capaz de exibir corretamente a imagem, levando a uma situação potencialmente médico-legal. Na Figura 2.3 é possível visualizar uma representação de como a estrutura dos arquivos DICOM é dividida (VARMA, 2012).

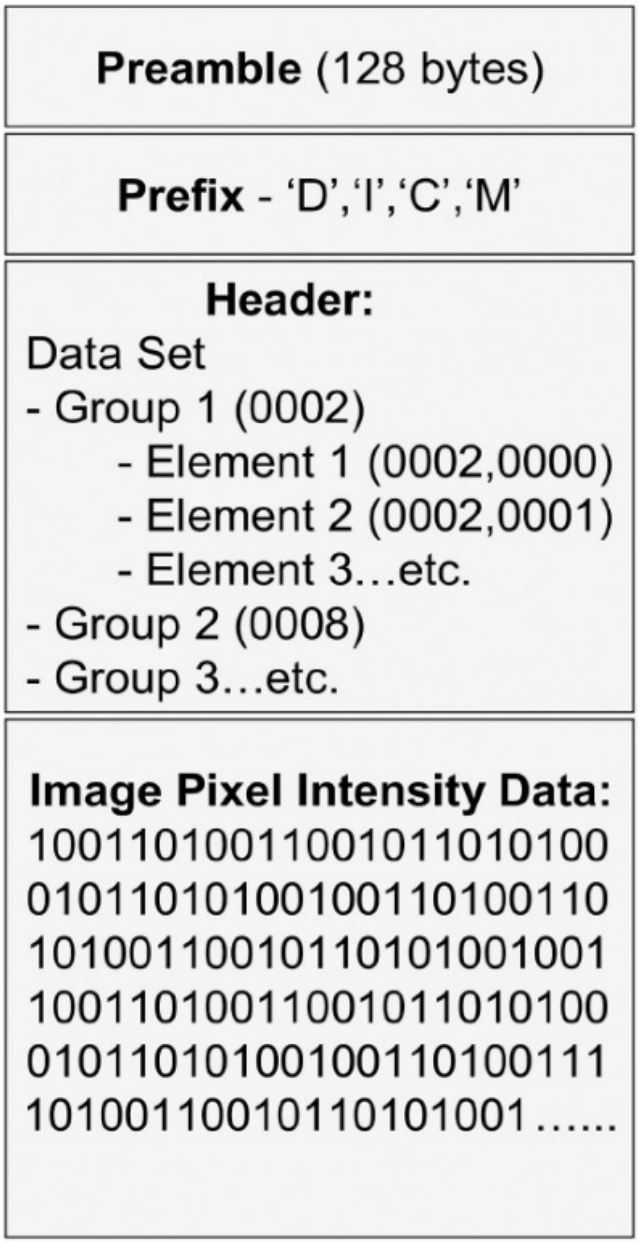

Figura 2.3. Estrutura de um arquivo .dcm que mostra o que cada parte desse tipo de arquivo contém, como cabeçalho com informações médicas gerais e os dados da imagem armazenada no arquivo (VARMA, 2012).

Para este trabalho, apenas as partes de imagens dos arquivos .dcm foram utilizadas para seu desenvolvimento.

2.3 *DATASET DUKEDBT*

O *dataset* utilizado neste trabalho é baseado em um conjunto de dados selecionados de imagens de tomossíntese digital de mama que inclui casos normais, *actionable*, benignos comprovados por biópsia e câncer comprovado por biópsia (BUDA, 2024).

2.3.1 Tipos de diagnósitcos no *dataset*

Diagnósticos *actionable* são aqueles em que são necessários maiores análises nos mesmos a fim de definir qual seria a classificação apropriada (BUDA, 2024).

Um tumor benigno é aquele que permanece em seu local primário sem invadir outros lugares do corpo. Ele não se espalha para estruturas locais ou para partes distantes do corpo, mas podem crescer lentamente. Eles geralmente não são problemáticos, mas podem crescer bastante e comprimir estruturas próximas, causando dor ou outras complicações médicas (PATEL, 2020).

O tumor maligno (i.e. câncer propriamente dito) possui células que crescem de forma descontrolada e se espalham localmente e/ou para locais distantes. Eles se espalham através da corrente sanguínea ou do sistema linfático. Esse espalhamento é chamado de metástase. A metástase pode ocorrer em qualquer parte do corpo e é mais comumente encontrada no fígado, pulmões, cérebro e ossos (PATEL, 2020).

Um diagnóstico normal significa que o paciente está saudável e não precisa de tratamento.

Como mencionado anteriormente, os exames dessa base da dados possuem até 4 ângulos diferentes para cada paciente - LCC, LMLO, RCC e RMLO. Nos casos de tumor presente, apenas as vistas em que ele pode ser visualizado são incluídas (BUDA, 2024).

Cada um dos ângulos diferentes de cada imagem possui um número de canais diferente, os quais também são chamados de *slices*. Nos casos em que há tumor presente, a fatia em que os médicos o identificaram é especificada. Para o trabalho, apenas um dos canais de cada imagem é utilizado e qual deles é selecionado irá depender do diagnóstico atribuído.

2.3.2 Estrutura do *dataset*

Os conjuntos de teste, treino e validação são instalados separadamente pelo site em que o a base de dados está presente. Como são uma coleção de arquivos DICOM, eles ocupam uma quantidade grande de memória.

O conjunto de dados foi originalmente utilizado para o desafio *DBTex2*, que contém um total de 22.032 exames de tomossíntese mamária de 5.060 pacientes, os quais são divididos em:

- Teste - 1.721 exames (135,14 GB)
- Treinamento - 19.148 exames (1,42 TB)
- Validação - 1.163 exames (84,71 GB)

Além dos arquivos DICOM, cada conjunto também possui 3 arquivos .csvs relacionados de grande importância — um que indica as *paths* em que cada arquivo será salvo, outro que indica em qual diagnóstico o arquivo se encaixa e, por fim, um que indica não só as posições das caixas delimitadoras (i.e. *bounding boxes*) que marcam o câncer (que não foi o foco deste trabalho), mas também em qual das fatias das imagens o câncer está presente.

2.3.3 *Paper* do *dataset*

O *dataset* possui junto a ele um *paper* introdutório da natureza dos dados funcionam e como foram pré-processados, além de informações breves acerca da rede proposta que usa um modelo DenseNet para a detecção de tumores nas imagens (BUDA, 2021).

O *paper* possui foco na tarefa de segmentação de imagens, enquanto que este trabalho buscou a classificação das mamografias em seus diagnósticos corretos.

A partir do que foi apresentado acerca dos tipos de arquivos que foram utilizados no trabalho e como eles foram obtidos, pode-se agora introduzir os conceitos acerca de Redes Neurais Convolucionais (CNNs) e as técnicas usadas em conjunto com elas.

CAPÍTULO 3

APRENDIZADO SUPERVISIONADO COM REDES NEURAIS CONVOLUCIONAIS

O aprendizado supervisionado é um dos métodos comumente utilizados no aprendizado de máquina. Nele um modelo é treinado utilizando um conjunto de dados rotulados, ou seja, dados onde as respostas corretas são conhecidas. O objetivo é que o modelo aprenda a mapear entradas para saídas corretas com base nesses exemplos e que seja capaz de generalizar essa habilidade para dados novos e não vistos. Aplicações típicas incluem classificação de imagens, reconhecimento de fala e predição de séries temporais (NG, 2024).

As Redes Neurais Convolucionais (CNNs) são uma arquitetura específica de redes neurais artificiais, criadas para processar dados que possuem uma estrutura em grade, como imagens. Introduzidas inicialmente na década de 1980 (Neocognitron), as CNNs ganharam popularidade a partir de 2012 (AlexNet), ao demostrar um desempenho superior em tarefas de reconhecimento de imagem. Utilizando camadas convolucionais, essas redes conseguem capturar características espaciais e padrões locais das imagens de maneira eficiente, o que permite a construção de modelos que podem reconhecer objetos, rostos e realizar diagnósticos médicos com alta precisão (KUMAR, 2021).

Neste capítulo, serão apresentados os funcionamentos de aprendizado supervisionado e CNNs e como podem ser utilizados para resolver problemas complexos de visão computacional, especificamente de diagnóstico de imagens médicas de tomossíntese. Também serão apresentadas ideias importantes acerca de aprendizado por transferência (i.e. *transfer learning*) e as arquiteturas utilizadas para este trabalho.

3.1 REDES NEURAIS CONVOLUCIONAIS

Redes Neurais Convolucionais (CNNs) são um tipo de Rede Neural Artificial (ANN) que possuem alta capacidade em reconhecimento de padrões. Por esse motivo, são usadas principalmente para aplicações em visão computacional como detecção de objetos e classificação de imagens. Uma arquitetura de uma CNN simples para classificação de dígitos é representada na Figura 3.1. Nota-se que ela possui uma camada convolucional com uma não-linearidade (ReLU), uma camada de *pooling* seguida de uma camada completamente conectada, seguida de uma camada de saída com 10 possíveis valores, que representam os 10 possíveis dígitos que a rede pode classificar a imagem de entrada.

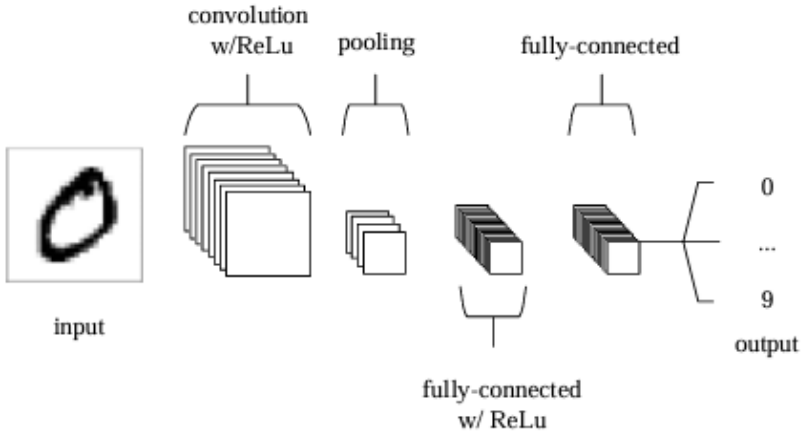

Figura 3.1. Arquitetura simples de uma CNN com apenas 5 camadas (O'SHEA; NASH, 2015).

3.1.1 Camadas convolucionais e filtros

O que principalmente diferencia as CNNs de ANNs padrão é o fato de que aquelas possuem camadas escondidas chamadas de camadas convolucionais, nas quais ocorre a operação de convolução. Assim como em ANNs, essas camadas possuem neurônios que recebem uma entrada, aplicam uma operação nela junto com uma não-linearidade e geram uma saída que será passada para frente da rede (O'SHEA; NASH, 2015).

As camadas convolucionas das CNNs possuem filtros, os quais também são chamados de *kernels*, que são responsáveis por aplicar a operação de convolução na entrada dos neurônios. São esses filtros que, em essência, são responsáveis pelo realce de padrões como bordas, formas

e objetos em imagens. De forma geral, as camadas convolucionais inicias são responsáveis por detectar padrões mais simples de imagens (e.g. bordas), enquanto que aquelas que estão mais profundas na rede reconhecem características mais complexas (e.g. objetos inteiros) (NG, 2024).

Os filtros das camadas convolucionais são matrizes de dimensão definida (geralmente 3x3, 5x5), em que cada uma de suas células tem um valor associado, que representam os pesos que serão responsáveis pela detecção de características nas imagens. De forma geral, os pesos dos *kernels* são inicializados aleatoriamente e atualizados durante um processo chamado *back propagation* (NG, 2024).

Os filtros associados às camadas convolucionais que aplicam a operação de convolução em suas entradas, sobrepondo pequenas partes das imagens definidas pela dimensão do *kernel*, começando do topo esquerdo, e deslizando-se aos poucos em quantidades definidas (i.e. *stride*) (NG, 2024).

A operação de convolução que o filtro aplica é definida por uma multiplicação de elemento por elemento na região que o *kernel* sobrepõe seguida de uma soma dos produtos, a qual pode ser visualizada na Figura 3.2 (REYNOLDS, 2019) (NG, 2024).

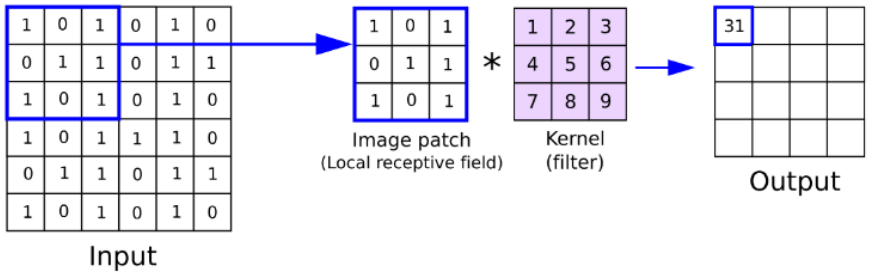

Figura 3.2. Representação do processo de convolução em CNNs. Na figura apenas a primeira etapa é representada, a qual gera o primeiro pixel da imagem de saída da camada. Para os pixels subsequentes, o mesmo processo é repetido. Na figura, o asterisco representa uma operação de produto interno entre o *patch* e o *kernel* (REYNOLDS, 2019).

A partir do apresentado na Figura 3.2, percebe-se que ocorre uma redução nas dimensões da saída da operação de convolução. Sem operações adicionais, a imagem resultante terá as seguintes dimensões (em um canal):

$$(n - f + 1) \times (n - f + 1) \tag{3.1}$$

em que n representa as dimensões originais da entrada e f representa as dimensões do filtro. É possível realizar operações conjuntas com a convolução para alterar a dimensão da imagem da saída, como *padding* e *stride*, que são apresentados a seguir (REYNOLDS, 2019) (NG, 2024).

3.1.2 *Padding*

Padding consiste em adicionar bordas de zeros ao redor da imagem de entrada da camada, o que, consequentemente, fará com que a saída fique com dimensões maiores. Além disso, essa operação irá incluir mais informações acerca das bordas da imagem original, visto que convoluções serão aplicadas mais vezes nelas (quando não há *padding* o filtro só vê, por exemplo, a borda esquerda superior da imagem uma vez) (NG, 2024).

A forma mais comum de aplicar *padding* é por meio da adição de zeros nas bordas da imagem de entrada (i.e. *zero-padding*), que é representada na Figura 3.3 (D, 2021) (NG, 2024).

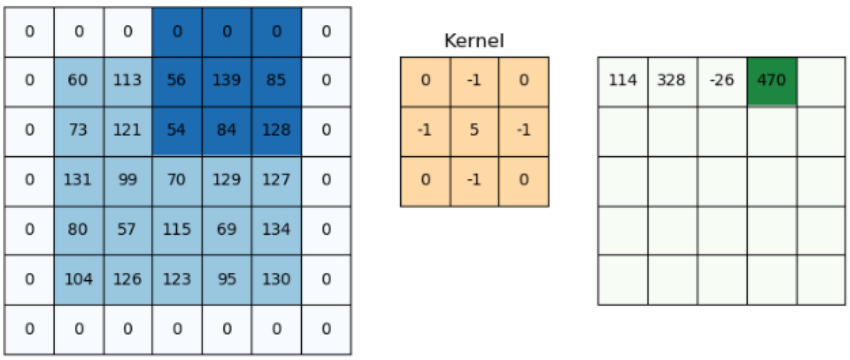

Figura 3.3. Exemplo de operação de convolução com zero-padding, usada quando não deseja-se que a imagem de saída de uma camada convolucional tenha suas dimensões reduzidas demais ou para incluir mais vezes os pixels das bordas na operação de convolução (D, 2021).

Uma entrada que teve uma convolução com padding terá a seguinte saída:

$$(n - f + 2p + 1) \times (n - f + 2p + 1) \tag{3.2}$$

em que p representa o número de bordas adicionadas via *padding*.

3.1.3 *Stride*

Além da operação de adição de bordas, também há a possibilidade de adição de *stride* nas camadas convolucionais, que consiste em quantos pixels o filtro irá deslizar pela imagem após uma convolução, tanto da esquerda para a direita como também de cima para baixo. Como o *kernel* irá pular partes da entrada, a saída resultante terá uma redução em suas dimensões (REYNOLDS, 2019) (NG, 2024).

A partir disso, a dimensão final de uma imagem após uma convolução tanto com *padding* como também com *stride* será:

$$\left(\frac{n-f+2p}{s}+1\right)\times\left(\frac{n-f+2p}{s}+1\right) \tag{3.3}$$

em que s representa quantos *pixels* serão pulados na operação de *stride* aplicada (REYNOLDS, 2019) (NG, 2024).

Muitas das imagens que são colocadas nas entradas de CNNs terão mais do que um canal (e.g. imagens PNG com canais RGB). O número de canais que o filtro que aplicará a convolução na imagem deve possuir o mesmo número de canais que ela. A operação será então realizada sobre todo o volume da entrada, que é representada na Figura 3.4 (REYNOLDS, 2019) (NG, 2024).

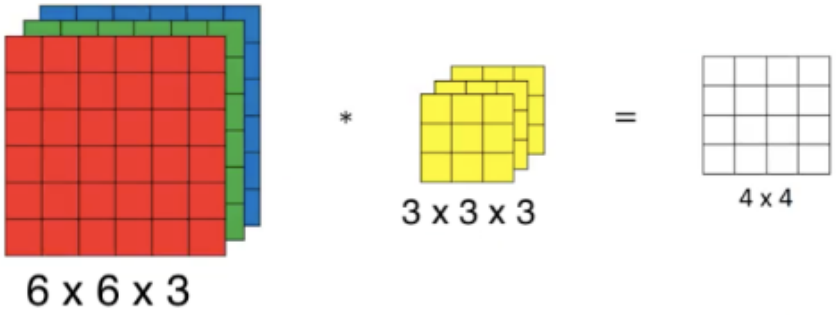

Figura 3.4. Representação de operação de convolução com os canais RGB da imagem representados. Nesta figura o asterisco representa a operação de convolução sendo aplicada (REYNOLDS, 2019).

Pode-se perceber da Figura 3.4 que cada canal do filtro irá aplicar uma convolução em cada respectivo canal da imagem e, ao final, o resultado é somado em apenas um canal na saída. Para aumentar o volume do resultado, deve-se utilizar um maior número de filtros, que resulta na seguinte dimensão para a saída:

$$\left(\frac{n-f+2p}{s}+1\right)\times\left(\frac{n-f+2p}{s}+1\right)\times n'_c \tag{3.4}$$

em que n'_c representa o número de filtros utilizados na camada. Considerando que o número de células dos filtros representam os pesos que serão treinados, tem-se então que a quantidade de parâmetros treinados em cada camada convolucional é dado por:

$$(f\times f\times n'_c+1)\times n_c \tag{3.5}$$

em que n_c representa o número de canais em cada filtro. Após a camada convolucional, é de praxe aplicar uma não-linearidade diretamente depois nos pesos e nos viéses (i.e. *biases*), muito comumente a *Rectified Linear Unit* (ReLU), que é uma função que tem valor zero para qualquer entrada negativa e para qualquer entrada positiva ela retorna o mesmo valor (NG, 2024).

3.1.4 *Pooling*

No geral, CNNs são compostas não só das camadas convolucionais, mas também de camadas de *pooling* e camadas completamente conectadas (FCL). Cada uma dessas camadas possui uma função específica na rede (NG, 2024).

Como mencionado anteriormente, camadas convolucionais serão responsáveis pela detecção dos padrões na imagem de entrada da rede por meio de filtros, comumente seguida de uma não-linearidade (i.e. função de ativação), principalmente a *Rectified Linear Unit* (ReLU) (NG, 2024).

Camadas de *pooling* são responsáveis por realizar uma redução na resolução da imagem de entrada (i.e. *down-sampling*) a fim de diminuir o número de parâmetros das operações e destacar determinadas características das imagens. Dentre os tipos operações de *pooling*, pode-se destacar (SAVYAKHOSLA, 2023):

- *Max Pooling* - Pega o valor máximo dentro de uma região definida por um filtro, geralmente com *stride*. Usado para destacar as características mais proeminetes da região, assim como visto na Figura 3.5 (SAVYAKHOSLA, 2023).

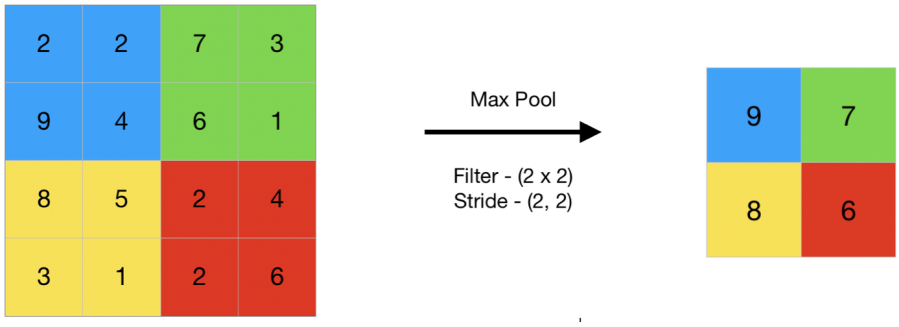

Figura 3.5. Representação de *Max Pooling* com filtro 2x2 e um stride de 2 (SAVYAKHOSLA, 2023).

- *Average Pooling* - Calcula o valor médio dos elementos presentes dentro da região de um filtro, geralmente com *stride*. Destaca a média das características dentro da região do filtro. Essa operação é representada na Figura 3.6 (SAVYAKHOSLA, 2023).

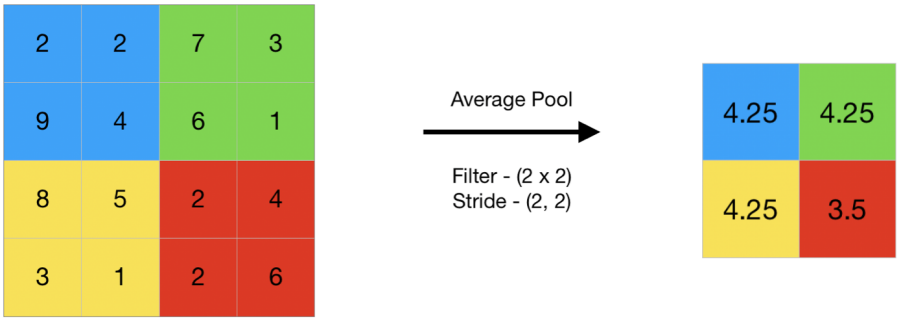

Figura 3.6. Representação de *Average Pooling* com filtro 2x2 e *stride* de 2 (SAVYAKHOSLA, 2023).

- *Global Pooling* - Reduz cada canal de uma entrada para um valor único. Um imagem de dimensões $n_h \times n_w \times n_c$ após esse tipo de pooling fica com dimensões $1 \times 1 \times n_c$. Comunmente usado para pegar ou o valor máximo ou valor médio de cada um dos canais (NG, 2024).

3.1.5 Camadas completamente conectadas

Camadas completamente conectadas são um tipo de camada de rede neural onde cada neurônio está conectado a todos os neurônios da camada anterior. Isso significa que todas as características de entrada são usadas para calcular cada característica de saída (NG, 2024).

Cada neurônio em uma camada totalmente conectada calcula uma soma ponderada de suas entradas, adiciona o viés e então aplica uma função de ativação. A representação matemática para uma camada totalmente conectada pode ser representada como:

$$y = f(\omega x + b) \tag{3.6}$$

em que y é um vetor de saída da camada que contém informações de cada neurônio, f representa a função de ativação usada, ω é a matriz de pesos, x é o vetor das entradas da camada e b é um vetor dos viéses (NG, 2024).

Camadas completamente conectadas são responsáveis pela interpretação das características reconhecidas pelas camadas convolucionais. Geralmente são colocadas mais ao final das redes para auxiliar na classificação final junto com uma camada *softmax* se houver mais de duas possíveis classes, caso contrário é comum utilizar uma saída com função de ativação sigmoide para classificação binária (O'SHEA; NASH, 2015) (NG, 2024).

Uma representação de uma FCC simples é mostrada na Figura 3.7. Destaca-se como nesses tipos de camadas cada neurônio é conectado com cada um dos neurônios das camadas subsequentes.

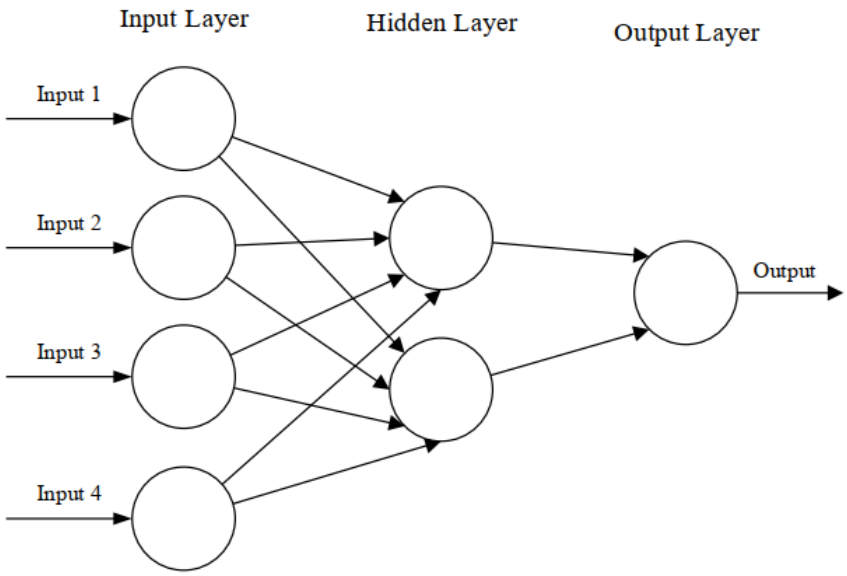

Figura 3.7. Representação de uma FCL com uma camada de entrada, sua camada escondida de 2 neurônios e uma saída (O'SHEA; NASH, 2015).

3.2 APRENDIZADO EM CNNS

Assim como em ANNs, CNNs utilizam técnicas como descida por gradiente (i.e. *gradient descent*), *forward propagation* (i.e. *forwardprop*) e *back propagation* (i.e. *backprop*), as quais utilizam funções de custo e otimizadores associados.

3.2.1 *Forward propagation*

Forwardprop é um processo em que os dados de entrada são processados através das camadas de uma rede neural para computar a saída da rede. Os dados de entrada passam pelas camadas da rede, com cada camada realizando operações como transformações lineares (somas ponderadas) e funções de ativação para gerar valores para as camadas subsequentes (NG, 2024).

O *forwardprop* é um processo que é comumente inicializado com pesos com valores aleatórios e viéses com zeros (NG, 2024). No caso de aprendizado transferido (*transfer learning*), pode-se ou utilizar os pesos já treinados da rede original ou retreiná-los por completo ou parcialmente.

Após cada etapa de *forwardprop*, ocorrerá subsequentemente o *backprop*, o qual depende da descida por gradiente e de uma função de custo.

3.2.2 Funções de perda e de custo

Quando trabalha-se com redes neurais, são utilizadas funções de perda e de custo para que seja possível o aprendizado da rede. A última camada terá como saída valores calculados de, por exemplo, uma classificação. A função de perda será responsável por realizar a comparação entre o valor calculado e o valor real da entrada da rede. A função de custo é a média de todas as perdas dos exemplos usados no treinamento da rede (NG, 2024).

Busca-se então minimizar o valor da função de custo a partir de ajustes nos pesos e vieses, o que resulta na maior proximidade naquilo que a rede prevê e os valores reais de sua entrada (LECUN, 2000).

Duas funções comumente utilizadas são a de entropia cruzada categórica para problemas multi-classe e entropia cruzada binária para problemas de classes binárias. Respectivamente,

são definidas, para um único exemplo, por:

$$\mathcal{L} = -\sum_{i=1}^{K} y_i \log(\hat{y}_i) \tag{3.7}$$

$$\mathcal{L} = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})] \tag{3.8}$$

em que K representa o número de classes, y representa o valor real e $\hat{y}$ representa o valor previsto pela rede. Para que os pesos sejam atualizados e ocorra uma melhoria no resultado de função de perda, utiliza-se em conjunto o *forwardprop* com o *backprop* para a descida por gradiente (NG, 2024).

3.2.3 Descida por gradiente e *backprop*

O aprendizado por gradiente baseia-se no fato de que geralmente é muito mais fácil minimizar uma função contínua e razoavelmente suave do que uma função discreta (combinatória). A função de custo pode ser minimizada estimando o impacto de pequenas variações nos valores dos parâmetros na função de perda (pesos e vises). Isso é medido pelo gradiente da função de perda em relação aos parâmetros (LECUN, 1998).

Algoritmos de aprendizado eficientes podem ser elaborados quando o vetor gradiente pode ser calculado de forma analítica (em oposição ao cálculo numérico através de perturbações). Esta é a base de numerosos algoritmos de aprendizado baseados em gradiente com parâmetros de valores contínuos (LECUN, 1998).

O *backprop* consiste em atualizar os pesos e vieses a partir do negativo do gradiente da funções de custo, visto que ele apontará para um possível mínimo da função. No caso de CNNs, os pesos e viés são matrizes que podem ter diversas dimensões, visto que dependem dos fatores mencionados anteriormente. Uma representação da descida de gradiente é representada na Figura 3.8 (GUDIMALLA, 2021) (NG, 2024).

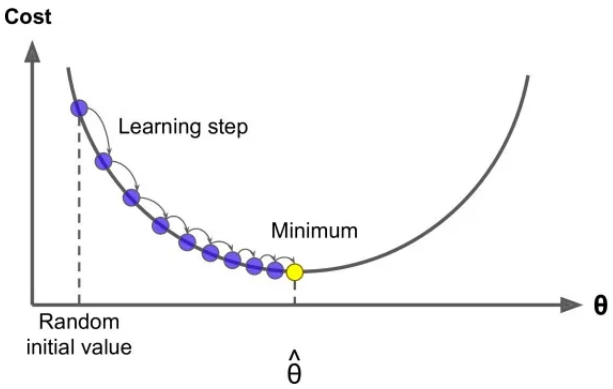

Figura 3.8. Visualização de descida por gradiente com inicialização aleatória de valores. A figura mostra como o valor da função de custo irá reduzir até que chegar em um valor mínimo, ou seja, convergir (GUDIMALLA, 2021).

O grau em que os valores dos pesos e dos vieses são atualizados dependem de um parâmetro chamado taxa de aprendizagem (i.e. *learning rate*). As formas com que esses parâmetros são atualizados são definidos pelo otimizadores aplicados (LECUN, 1998) (NG, 2024). Antes de falar deles, convém diferenciar os conceitos de aprendizado estocástico e aprendizado por lotes (i.e. *batches*).

3.2.4 Tipos de aprendizado

De forma geral, existem três tipos principais de formas em que uma rede neural pode aprender; aprendizado estocástico, aprendizado em *batches* e aprendizado por *mini-batches*.

Deep Learning tende a apresentar melhor desempenho quando treinado com grandes quantidades de dados, o que pode resultar em tempos de treinamento muito longos. Por esse motivo, a escolha de algoritmos eficientes é importante para lidar com esse grande volume de dados e otimizar o processo de treinamento (NG, 2024). Durante o treinamento da rede, o modelo analisa repetidamente os mesmos dados de um conjunto de treinamento, passando por várias iterações conhecidas como épocas (i.e. *epochs*). Cada época representa uma passagem completa por todo o conjunto de dados de treinamento, o que permite que o modelo ajuste seus parâmetros e melhore seu desempenho a cada ciclo.

Ao utilizar uma base de dados com 5 milhões de exemplos de treinamento, se a rede analisar todos eles de uma vez pode levar uma quantidade muito grande de tempo para ocorrer a atualização de parâmetros. Por outro lado, a função de custo irá ser reduzida a cada época de

forma suave, o que deixa as condições de convergência mais claras e as análise de atualização de pesos e viéses mais simples (LECUN, 2000).

Esse processo em que a rede visualiza o *dataset* por completo antes de atualizar os parâmetros é chamado de aprendizado por *batches*, que é representado na Figura 3.9 (NG, 2024).

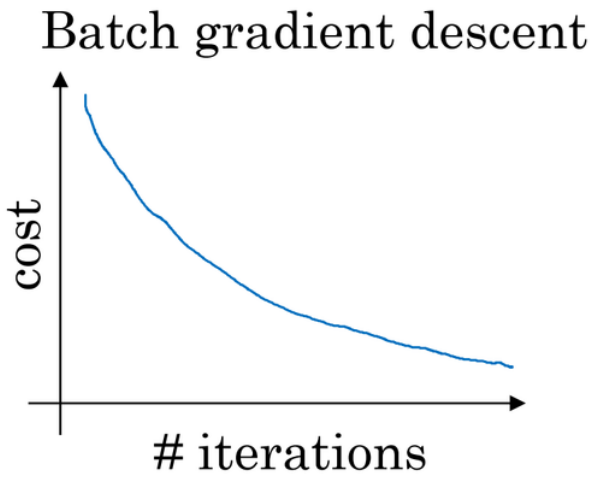

Figura 3.9. Representação de descida por gradiente em *batch*. É notável como a função de custo é reduzida por época de forma suave até eventualmente chegar em um mínimo (NG, 2024).

Por outro lado, existe o treinamento estocástico, que consiste em atualizar os parâmetros da rede a cada exemplo que ela visualiza. Esse tipo de aprendizado tende a ser muito mais rápido que o aprendedizado em *batch*, resulta em soluções melhores e permite uma visualização rápida das mudanças que ocorrem no treinamento (LECUN, 2000). Apesar disso, essa técnica é extramemente ruidosa e pode não atingir um verdadeiro mínimo global, apenas ficará flutuando em torno dele (NG, 2024).

Por fim, existe o treinamento em *mini-batches*, o qual consiste em separar o conjunto de treinamento em pacotes menores, o qual a rede irá analisar e atualizar seus parâmetros ao visualizar por inteiro cada um deles. Isso torna esse processo um meio termo entre o aprendizado estocástico e o em *batch*. Ele será um pouco mais ruidoso e rápido que o segundo, mas diferente do primeiro possui maior probabilidade de atingir um mínimo global e, por esses motivos, é mais comumente utilizado (NG, 2024). Esse tipo de aprendizado poder visualizado na Figura 3.10.

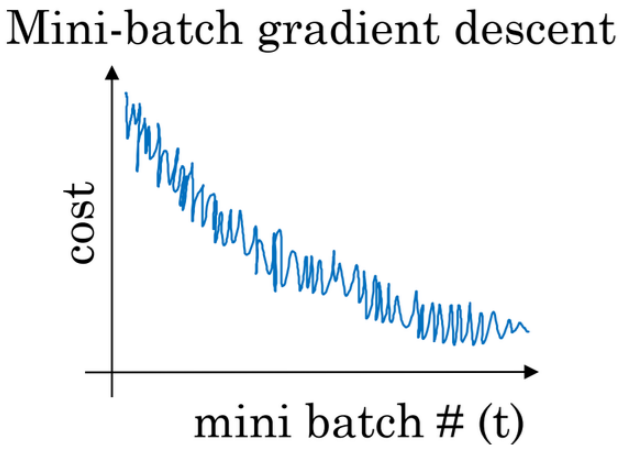

Figura 3.10. Representação de descida por gradiente em *mini-batch*. Percebe-se maior ruidosidade no exemplo, mas mesmo assim ele é capaz de atingir um mínimo global (NG, 2024).

Com o conceito desses tipos de aprendizados em mente, pode-se agora introduzir o conceito de otimizadores em redes neurais.

3.2.5 Otimizadores

Uma das formas mais simples de atualizar os valores dos parâmetros é por meio da descida por gradiente estocástica (SGD), que é definida por:

$$\omega_k = \omega_{k-1} - \alpha \frac{\partial J(\omega, b)}{\partial \omega} \tag{3.9}$$

$$b_k = b_{k-1} - \alpha \frac{\partial J(\omega, b)}{\partial b} \tag{3.10}$$

em que $J(\omega, b)$ representa a função de custo utilizada pela rede, ω representa a matriz de pesos, b é a matriz de viéses e α representa a taxa de aprendizado. O subíndice k representa em qual iteração de *mini-batch* a rede se encontra (LECUN, 1998).

Para facilitar a notação, as derivadas parciais dos pesos e vieses serão representadas agora em diante como $d\omega$ e db.

Uma outra forma comum de otimizar os pesos e vieses de uma rede é por meio do uso de uma descida por gradiente com momento, que é definida por deixar o processo de aprendizado menos ruidoso e pode facilitar o processo de encontrar um mínimo global:

$$V_{d\omega,k} = \beta V_{d\omega,k-1} + (1-\beta)d\omega \tag{3.11}$$

$$V_{db,k} = \beta V_{db,k-1} + (1-\beta)db \tag{3.12}$$

$$\omega_k = W_{k-1} - \alpha V_{d\omega,k} \tag{3.13}$$

$$b_k = b_{k-1} - \alpha V_{db,k} \tag{3.14}$$

em que β representa o coeficiente de momento, o qual pode ser variado. No geral, utilizar $\beta = 0{,}9$ traz bons resultados para o otimizador (NG, 2024).

Além desses otimizadores, existe ainda um alternativo chamado *Root Mean Square propagation* (RMSprop), o qual é caracterizado por, assim como a descida por gradiente com momento, acelerar o processo de encontrar um mínimo global (NG, 2024).

O algorítmo de RMSprop é baseado na atualização de parâmetros por meio da inclusão de uma divisão que leva em consideração médias ponderadas exponecialmente (i.e. *exponentially weighted averages*):

$$S_{d\omega,k} = \beta S_{d\omega,k-1} + (1-\beta)d\omega^2 \tag{3.15}$$

$$S_{db,k} = \beta S_{db,k-1} + (1-\beta)db^2 \tag{3.16}$$

$$\omega_k = \omega_{k-1} - \alpha \frac{d\omega}{\epsilon + \sqrt{S_{d\omega}}} \tag{3.17}$$

$$b_k = b_{k-1} - \alpha \frac{db}{\epsilon + \sqrt{S_{db}}} \tag{3.18}$$

em que o β do RMSprop não possui a mesma definição que na descida por gradiente por momento. Aqui ele representa um parâmetro de média móvel (i.e. *moving average*). O ϵ é um

valor muito baixo que existe para evitar divisões por zero, o qual geralmente possui valor fixo de 10^{-8} (NG, 2024).

Por fim, um outro otimizador muito utilizado é o Adam, o qual, em essência, junta RMSprop e descida de gradiente com momento:

$$V_{d\omega,k} = \beta_1 V_{d\omega,k-1} + (1-\beta_1) d\omega \tag{3.19}$$

$$V_{db,k} = \beta_1 V_{db,k-1} + (1-\beta_1) db \tag{3.20}$$

$$S_{d\omega,k} = \beta_2 S_{d\omega,k-1} + (1-\beta_2) d\omega^2 \tag{3.21}$$

$$S_{db,k} = \beta_2 S_{db,k-1} + (1-\beta_2) db^2 \tag{3.22}$$

$$V_{d\omega}^{Corrigido} = \frac{V_{d\omega}}{1-\beta_1^t} \tag{3.23}$$

$$V_{db}^{Corrigido} = \frac{V_{db}}{1-\beta_1^t} \tag{3.24}$$

$$S_{d\omega}^{Corrigido} = \frac{S_{d\omega}}{1-\beta_2^t} \tag{3.25}$$

$$S_{db}^{Corrigido} = \frac{S_{db}}{1-\beta_2^t} \tag{3.26}$$

$$\omega_k = \omega_{k-1} - \alpha \frac{V_{d\omega}^{Corrigido}}{\epsilon + \sqrt{S_{d\omega}^{Corrigido}}} \tag{3.27}$$

$$b_k = b_{k-1} - \alpha \frac{V_{db}^{Corrigido}}{\epsilon + \sqrt{S_{db}^{Corrigido}}} \tag{3.28}$$

em que α é a taxa de aprendizagem, β_1 é p coeficiente de momento (i.e. chamado de primeiro momento), β_2 é a média móvel (i.e. segundo momento), ϵ é um valor fixo (10^{-8}) e t representa

a atual iteração do *mini-batch*. Os valores com superíndice *Corrigido* são termos de correção de viés (i.e. *bias correction*), que é uma técnica usada para ajustar a estimativa de momento e de média móvel durante a inicialização do algorítimo de *backprop* (NG, 2024).

3.3 FORMAS DE AVALIAÇÃO APÓS O TREINAMENTO

Para avaliar a eficácia do treinamento da rede neural, são utilizadas diferentes métricas e cálculos baseados nos resultados obtidos ao aplicar o modelo na base de teste. Essas métricas permitem uma análise detalhada do desempenho do modelo e ajudam a identificar pontos fortes e áreas que necessitam de melhorias (NG, 2024).

As métricas de avaliação geralmente baseiam-se nos conceitos de verdadeiros positivos (*True Positives* - TP), verdadeiros negativos (*True Negatives* - TN), falsos positivos (*False Positives* - FP), falsos negativos (*False Negatives* - FN), positivos (*Positives* - P) e negativos (*Negatives* - N) (NG, 2024):

- *True Positives* (TP) - São os casos em que o modelo previu corretamente um exemplo da classe A como pertencente à classe A. Em um problema de classificação binária, se a classe positiva representa a presença de uma doença, um verdadeiro positivo seria um paciente que tem a doença e foi corretamente identificado pelo modelo.
- *True Negatives* (TN) - São os casos em que o modelo acertou ao não classificar um exemplo da classe A como não pertencente à classe B. Um verdadeiro negativo seria um paciente que não tem a doença e foi corretamente identificado como não tendo a doença pelo modelo.
- *False Positives* (FP) - São os casos em que o modelo previu incorretamente um exemplo de uma outra classe B como pertencente à classe A. Um falso positivo seria um paciente que não tem a doença, mas o modelo previu que ele tem.
- *False Negatives* (FN) - São os casos em que o modelo previu incorretamente um exemplo da classe A como pertencente a uma outra classe B. Um falso negativo seria um paciente que tem a doença, mas o modelo previu que ele não tem.
- *Positives* (P) - São todos os casos que pertencem a uma determinada classe ($TP + FN$).

- *Negatives* (N) - São todos os casos que não pertencem a uma determinada classe ($TN + FP$).

3.3.1 Acurácia

A acurácia é uma medida da proporção de previsões corretas em relação ao total de previsões. Ela pode ser calculada a partir de (NG, 2024):

$$\frac{TP + TN}{P + N} \tag{3.29}$$

3.3.2 Especificidade

A especificidade é responsável por medir a proporção de verdadeiros negativos em relação ao total de instâncias que realmente pertencem à classe negativa, ou seja, é a capacidade do modelo de identificar corretamente as instâncias negativa, que pode ser calculada como (NG, 2024):

$$\frac{TN}{TN + FP} \tag{3.30}$$

3.3.3 Sensibilidade

A sensibilidade mede a proporção de verdadeiros positivos em relação ao total de instâncias que realmente pertencem à classe positiva. Em outras palavras, é a capacidade do modelo de identificar corretamente as instâncias positivas. Ela representa a métrica mais importante para esse trabalho, visto que seu objetivo é a detecção correta de casos de câncer. Essa grandeza é calculada a partir de (NG, 2024):

$$\frac{TP}{TP + FN} \tag{3.31}$$

3.3.4 Precisão

A precisão é uma métrica que mede a proporção entre os verdadeiros positivos em relação ao todal de classificações verdadeiros positivos e falsos positivos, ou seja (NG, 2024):

$$\frac{TP}{TP + FP} \tag{3.32}$$

3.3.5 *F1-score*

O *F1-score* representa a média harmônica da precisão e da sensibilidade. A média harmônica é utilizada pois penaliza valores extremos, o que garante que um baixo valor de precisão ou sensibilidade resulte em um *F1-score* baixo. Essa grandeza pode ser calculada a partir de (NG, 2024):

$$\frac{2TP}{2TP + FP + FN} \tag{3.33}$$

3.4 TÉCNICAS DE REGULARIZAÇÃO

Técnicas de regularização são métodos usados para prevenir o *overfitting*, ou seja, para melhorar a capacidade do modelo de generalizar bem para dados não vistos pela rede. O *overfitting* ocorre quando o modelo se ajusta muito bem aos dados de treinamento, mas não consegue ter um bom desempenho em novos dados. As técnicas de regularização ajudam a controlar a complexidade do modelo e aprender representações mais robustas (NG, 2024).

3.4.1 Regularização L2

A regularização L2 funciona a partir da adição de um termo de penalidade (λ) à função de perda que o modelo utiliza. Esse termo de penalidade é proporcional à soma dos valores quadrados de todos os pesos no modelo. Ele possui o objetivo de forçar o modelo a ter pesos com valores baixos, o que torna-o mais simples e com menor probabildade de sofrer *overfitting*

(NG, 2024).

O termo de penalidade é um hiperparâmetro que pode ser definido manualmente e, no geral, requer ajustes finos para encontrar o valor ótimo para determinada tarefa (NG, 2024).

Levando-se em consideração uma função de custo genérica com regularização L2 aplicada, tem-se que:

$$J(\omega, b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}||\omega||_2^2 \tag{3.34}$$

$$||\omega||_2^2 = \sum_{j=1}^{\eta_x}\omega_j^2 = \omega^T\omega \tag{3.35}$$

em que m representa o número de exemplos do *dataset*, i representa o i-ésimo exemplo de treinamento, λ é o termo de regularizador, $||\omega||_2^2$ é a norma euclidiana do peso, que representa a soma ao quadrado de todos os pesos e η_x representa o número total de pesos (NG, 2024).

Ao considerar a as matrizes de pesos utilizadas em redes neurais, tem-se então que:

$$J(\omega^{[L]}, b^{[L]}) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L}||\omega^{[l]}||_2^2 \tag{3.36}$$

$$||\omega^{[l]}||_2^2 = \sum_{i=1}^{\eta^{[l-1]}}\sum_{j=1}^{\eta^{[l]}}(\omega_{ij}^{[l]})^2 \tag{3.37}$$

$$\omega^{[l]} = \sqrt{\sum_{i=1}^{\eta^{[l-1]}}\sum_{j=1}^{\eta^{[l]}}(\omega_{ij}^{[l]})^2} \tag{3.38}$$

em que L representa o número total de camadas da rede, l representa a camada atual e a matriz da norma dos pesos é chamada de norma de Forbenius (NG, 2024).

Por fim, nesse tipo de regularização os pesos serão então atualizados da seguintes forma:

$$d\omega^{[l]} = b.p. + \frac{\lambda}{m}\omega^{[l]} \tag{3.39}$$

$$\omega^{[l]} = \omega^{[l]} - \alpha d\omega^{[l]} \tag{3.40}$$

em que *b.p.* é o valor resultante do *back propagation*. A regularização L2 é também chamada de *weight decay* pois força com que o os pesos tenham valores menores devido ao termo de regularização (NG, 2024).

3.4.2 *Dropout*

A técnica de regularização de *dropout* desativa aleatoriamente uma fração de neurônios (exceto os da camada de saída) na rede em cada iteração. Esses neurônios desativados são ignorados tanto durante a etapa de *forwardprop* quanto durante a etapa de *back propagation*. A probabilidade de desativação é determinada por um hiperparâmetro chamado taxa de *dropout*. É importante ressaltar que esse processo é realizado apenas na etapa de treinamento da rede (NG, 2024).

Esse tipo de regularização funciona porque evita que a rede se concentre excessivamente em determinadas características, forçando-a a distribuir os pesos de forma mais uniforme. Esse processo pode ser visualizado na Figura 3.11.

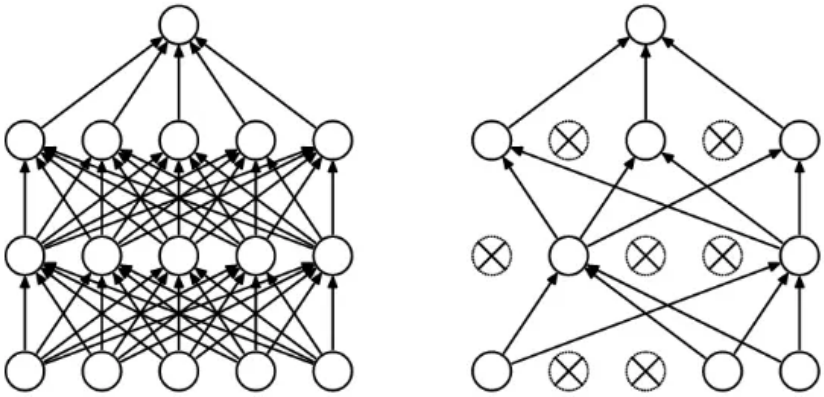

Figura 3.11. Representação de *dropout* em redes neurais, em que, durante o treinamente, neurônios aleatórios da rede são desativados a fim de reduzir *overfitting* (NG, 2024).

3.4.3 *Early stopping*

Early stopping é uma técnica de regularização que consiste em interromper o treinamento antes que o modelo comece a memorizar os dados do conjunto de treinamento e perca desempenho nos dados de validação (NG, 2024).

Esse método envolve o uso de um critério de parada chamado paciência, o qual monitora continuamente a perda de validação por um número determinado de épocas. Se a perda de validação não melhorar, o treinamento é interrompido e os pesos da época com a melhor perda de validação são recuperados e salvos no modelo para uso nos testes (NG, 2024).

Em suma, o *early stopping* é utilizado para garantir que o modelo treinado tenha uma melhor capacidade de generalização e não memorize os dados de treinamento. O processo de interrupção do treino pode ser visto na Figura 3.12.

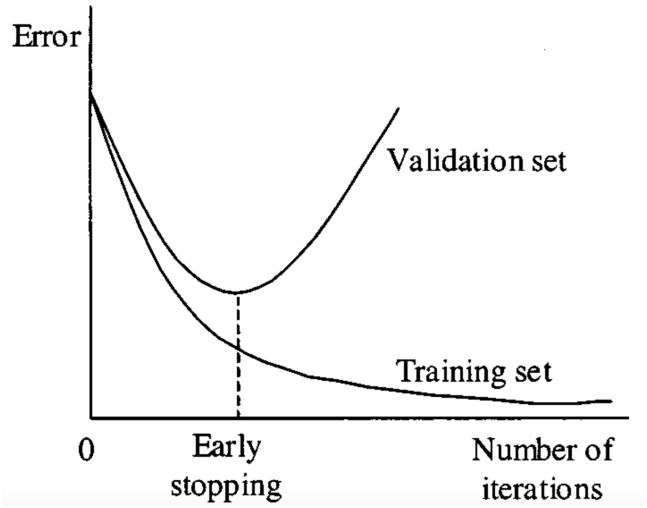

Figura 3.12. Representação de *early stopping*, em que o treinamento é interrompido se a função de perda de validação não melhorar após um número pré-definido de épocas consecutivas, conhecido como paciencia (GENCAY, 2021) (NG, 2024).

3.4.4 *Data augmentation*

A técnica de regularização de *data augmentation* envolve a criação de novas amostras de dados de treinamento a partir de transformações dos dados existentes. Essas transformações podem incluir operações como rotações, translações, cortes, mudanças de escala, ajustes de brilho e contraste, adição de ruído, etc. O objetivo é aumentar a diversidade dos dados de treinamento e, assim, melhorar a capacidade de generalização do modelo (NG, 2024).

Essa forma de regularização é utilizada principalmente quando há poucos dados de treinamento que a rede possa utilizar ou quando há um desbalanceameneto muito grande de dados. Ela funciona porque expande o conjunto de treinamento de forma artificial, permitindo que a rede neural aprenda a reconhecer padrões mais gerais e robustos, o que ajuda a previnir o

overfitting (NG, 2024). Algumas operações de *data augmentation* podem ser visualizadas na Figura 3.13.

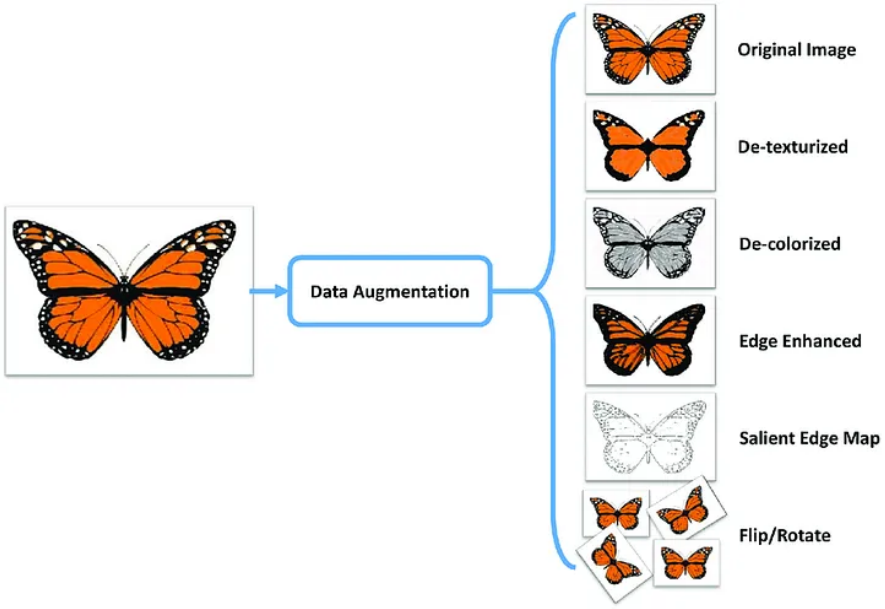

Figura 3.13. Representação de *data augmentation*, em que diferentes operações são aplicadas nas imagens de treinamento a fim de gerar uma maior base de dados artificial e reduzir *overfitting* (HOSNI, 2023) (NG, 2024).

3.4.5 Função de perda ponderada

O uso da função de perda ponderada em CNNs é uma técnica que modifica a função de perda com o objetivo de priorizar certas classes em detrimento de outros durante o treinamento. Assim como o *data augmentation*, ela é útil em casos de bases de dados desbalanceados, onde existem classes sub-representadas em relação às outras (NG, 2024).

Essa forma de regulariação consiste em adicionar pesos adicionais na função de perda, o que força com que erros em uma classe sejam mais custosos do que em outra (e.g. classificar um caso de câncer como normal). Em outras palavras, quando a rede cometer um erro mais custoso, o resultado da função de perda irá aumentar (NG, 2024).

Por exemplo, a perda de entropia cruzada binária ponderada será dada por:

$$\mathcal{L} = -[\omega_0 y_i \log(\hat{y}_i) + \omega_1 (1 - y_i) \log(1 - \hat{y}_i)] \tag{3.41}$$

$$\omega_i = \frac{N}{N_i \times n_{classes}} \tag{3.42}$$

em que ω_0 e ω_1 são os pesos atribuídos para cada classe, os quais são calculados a partir da Equação 4.42, na qual N representa o número total de exemplos, N_i representa o número de exemplos da i-ésima classe e $n_{classes}$ representa a quantidade de classes diferentes (NG, 2024).

3.5 TRANSFER LEARNING

Ao construir uma aplicação de visão computacional, em vez de treinar os pesos da rede do zero a partir de uma inicialização aleatória, frequentemente é possível obter um desempenho muito mais rápido (e melhor) ao utilizar pesos que foram previamente treinados por outra rede e usá-los em outra tarefa de interesse, processo chamado de *transfer learning* (NG, 2024).

É possível realizar o congelamento dos pesos e vieses das redes baixadas para que eles não sejam alterados durante o treinamento. Nesses casos, a última camada de classificação original é substituída por uma camada própria, específica para a nova aplicação. Dessa forma, apenas a nova camada será treinada do zero, enquanto o restante da rede mantém os conhecimentos adquiridos previamente. Isso é útil pois a rede original pode ter sido treinada para um propósito diferente ou possui um número de classes distinto (NG, 2024).

Congelar a rede inteira é geralmente feito quando não se tem uma quantidade de dados elevada. Quando a base de dados é grande, é comum descongelar alguns dos últimos *layers* da rede, permitindo que ela aprenda mais sobre os detalhes finos das novas entradas. O descongelamente gradual da rede também é comum como uma forma de fino ajuste, para melhorar seu desempenho (NG, 2024).

Além de descongelar aos poucos a rede, também é de praxe adicionar algumas camadas completamente conectadas após as originais e antes da nova camada de classificação, a fim de tentar extrair mais informações sobre as entradas (NG, 2024).

Existem diversas redes *open-source* que podem ser baixadas diretamente. Para este projeto, as principais utilizadas foram a DenseNet-121, ResNet-50 e a VGG-16.

3.5.1 DenseNet-121

Uma DenseNet é uma uma arquitetura que é definida por um padrão de conectividade simples a fim de garantir o fluxo máximo de informações entre as camadas na rede. Ela conecta todas as camadas (com tamanhos de mapas de características correspondentes) diretamente umas às outras. Para preservar a natureza de alimentação direta, cada camada obtém entradas adicionais de todas as camadas precedentes e passa seus próprios mapas de características para todas as camadas subsequentes. A Figura 3.14 ilustra uma camada densa de forma esquemática. (HUANG, 2016).

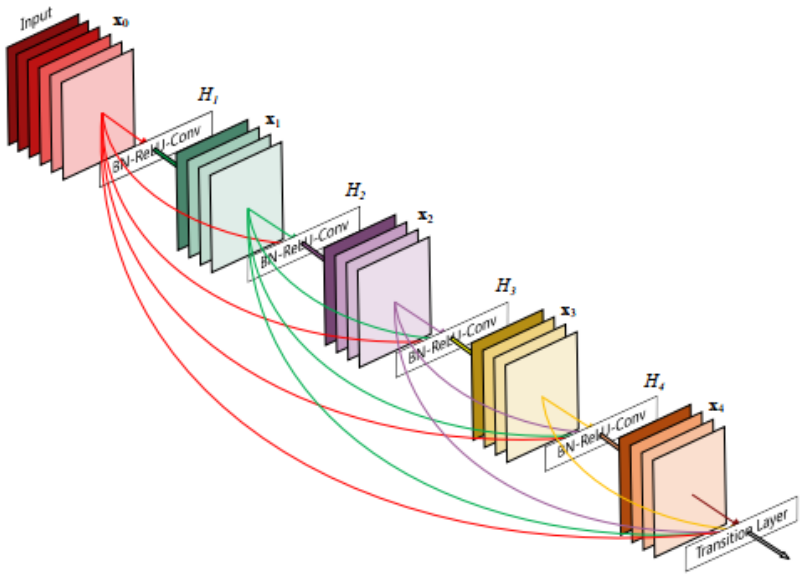

Figura 3.14. Um bloco de camada densa de 5 *layers* com uma taxa de crescimento de k = 4. Cada camada utiliza todos os mapas de características precedentes como entrada. (HUANG, 2016).

As DenseNets nunca combinam características por soma antes de serem passadas para uma camada. Em vez disso, elas juntam características por meio de concatenações. Assim, a camada n tem n entradas, as quais consistem nos mapas de características de todos os blocos de convolução precedentes. Seus próprios mapas de características são passados para todas as $L - n$ camadas subsequentes. Isso introduz $L(L + 1)/2$ conexões em uma rede de L camadas, em vez de apenas L, como nas arquiteturas tradicionais. A Figura 3.15 mostra uma arquitetura de uma DenseNet com 3 blocos densos (HUANG, 2016).

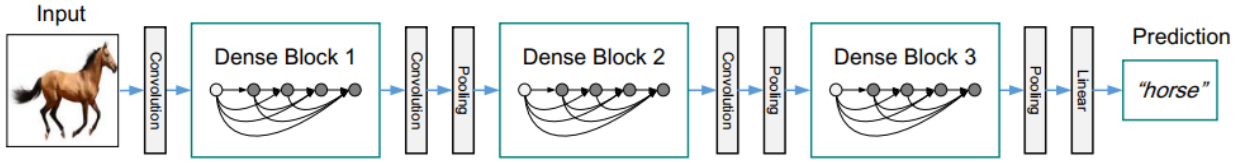

Figura 3.15. Um DenseNet com três blocos densos. As camadas entre dois blocos adjacentes são referidas como camadas de transição e alteram os tamanhos dos mapas de características via convolução e pooling. (HUANG, 2016).

DenseNets precisam de menos parâmetros do que as redes convolucionais tradicionais, pois não há necessidade de reaprender mapas de características redundantes. Arquiteturas tradicionais de alimentação direta podem ser vistas como algoritmos com um estado, que é passado de camada em camada. Cada camada lê o estado de sua camada precedente e escreve para a camada subsequente. Ela modifica o estado, mas também passa informações que precisam ser preservadas. Isso faz com que DenseNets sejam capazes de diferenciar explicitamente entre a informação que é adicionada à rede e a informação que é preservada (HUANG, 2016).

Além de melhor eficiência de parâmetros, uma grande vantagem das DenseNets é o fluxo aprimorado de informações e gradientes por toda a rede, o que as torna fáceis de treinar. Cada camada tem acesso direto aos gradientes da função de perda e ao sinal de entrada original. Isso ajuda no treinamento de arquiteturas de rede mais profundas. Além disso, conexões densas têm um efeito regularizador, que reduz o *overfitting* em tarefas com tamanhos menores de conjuntos de treinamento (HUANG, 2016).

A DenseNet-121 é uma das variações de uma DenseNet com 121 camadas em sua estrutura. Sua arquitetura é representada na Figura 3.16.

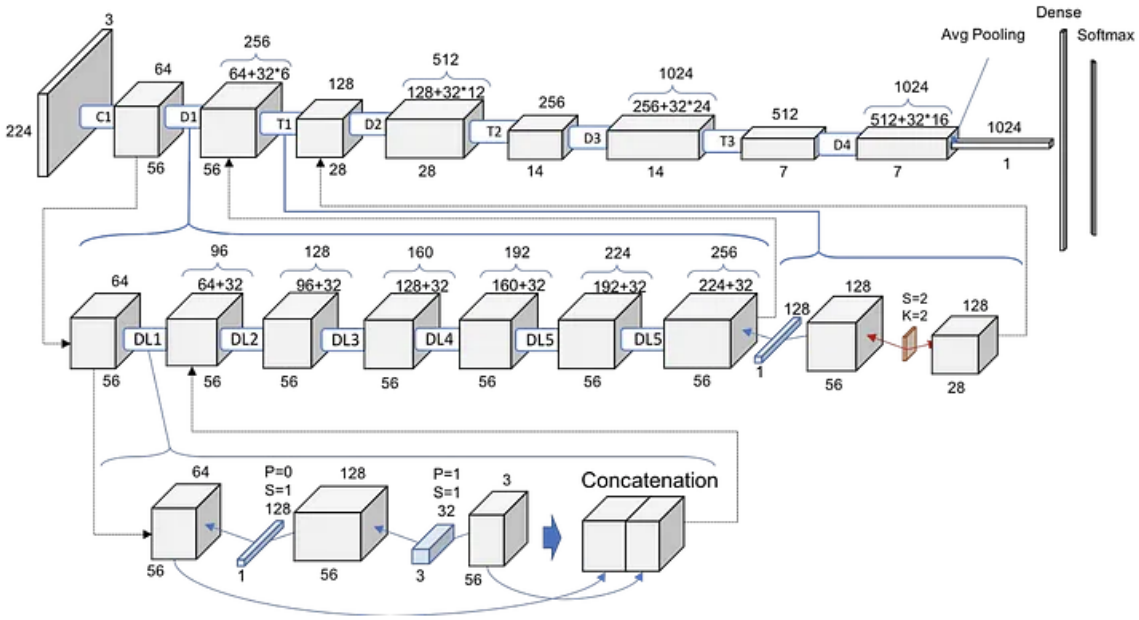

Figura 3.16. Arquitetura da DenseNet-121, que é uma variação de uma DenseNet com 121 camadas (PATEL, 2023).

3.5.2 ResNet-50

Redes muito profundas são difíceis de treinar devido aos problemas relacionados aos gradientes que explodem ou desaparecem.

O problema de gradientes que explodem ocorre quando os gradientes das camadas iniciais se tornam grandes durante o treinamento e crescem exponencialmente. Isso faz com que os pesos da rede sejam atualizados de forma excessiva, levando a instabilidades e a um desempenho ruim do modelo.

Já o problema de gradientes que desaparecem ocorre quando os gradientes das camadas iniciais de uma rede neural se tornam pequenos durante o treinamento. Como resultado, os pesos das camadas subsequentes não são atualizados de forma eficaz, o que prejudica o aprendizado da rede.

ResNets são caracterizadas por utilizarem um tipo de conexão especial denominadas *skip-connections*, que permitem que a rede colete uma ativação de uma determinada camada e alimentá-la para outra camada mais profunda na rede neural. Essas conexões permitem o treinamento de redes profundas de forma mais eficiente e mitigam problemas com gradientes (NG, 2024) (HE, 2015).

Em ResNets, as *skip-connections* são usadas em camadas chamadas de blocos residuais, os quais alimentam ativações anteriores diretamente en outra camada dentro de sua função de ativação não-linear. A Figura 3.17 mostra a representação dessa conexão.(NG, 2024) (HE, 2015)

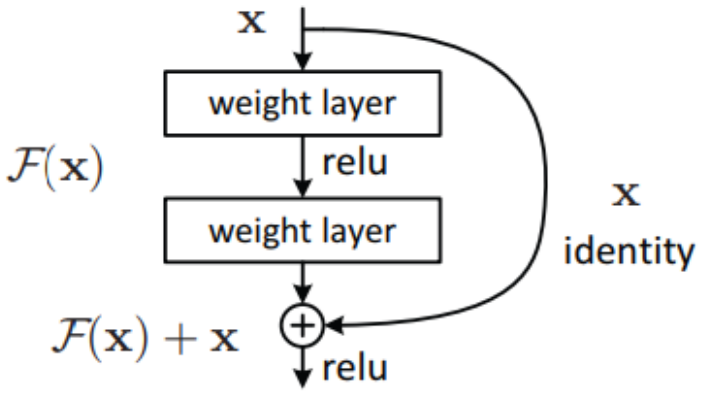

Figura 3.17. Representação de um bloco residual de uma ResNet, no qual a saída da função de ativação de uma camada é alimentada diretamente na não-linearidade de outro *layer* que está mais profundo na rede (HE, 2015).

Em essência, uma ResNet consiste em vários blocos residuais subsequentes. Um exemplo de uma ResNet de 34 camadas comparada com uma rede comum com a mesma quantidade de *layers* é mostrada na figura 3.18. A ResNet-50 é uma ResNet com 50 camadas.

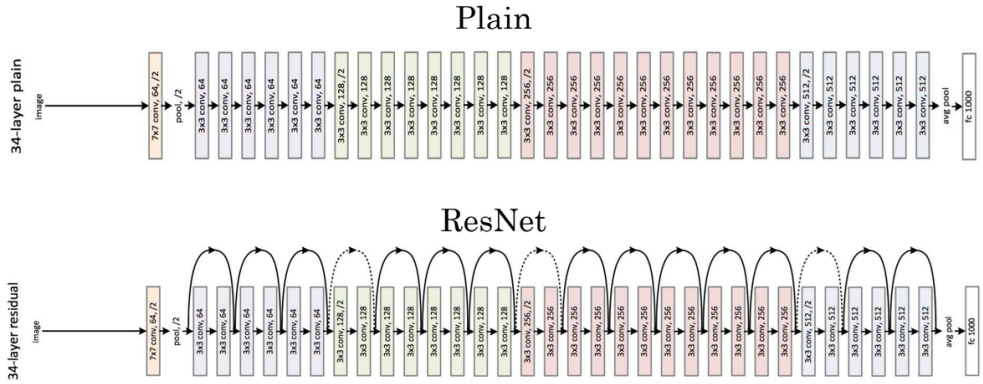

Figura 3.18. Comparação de uma rede comum de 34 camadas com uma ResNet com 34 blocos residuais (HE, 2015) (NG, 2024).

3.5.3 VGG-16

As redes VGG foram criadas por um grupo chamado *Virtual Geometry Group* (VGG) com a finalidade de testar como a profundidade de CNNs afetam o desempenho no treinamento. No caso da VGG-16, ela possui 16 camadas com pesos treináveis, o que inclui *layers* convolucionais e completamente conectados. Sua arquitetura é representada na Figura 3.19 (SIMONYAN, 2014).

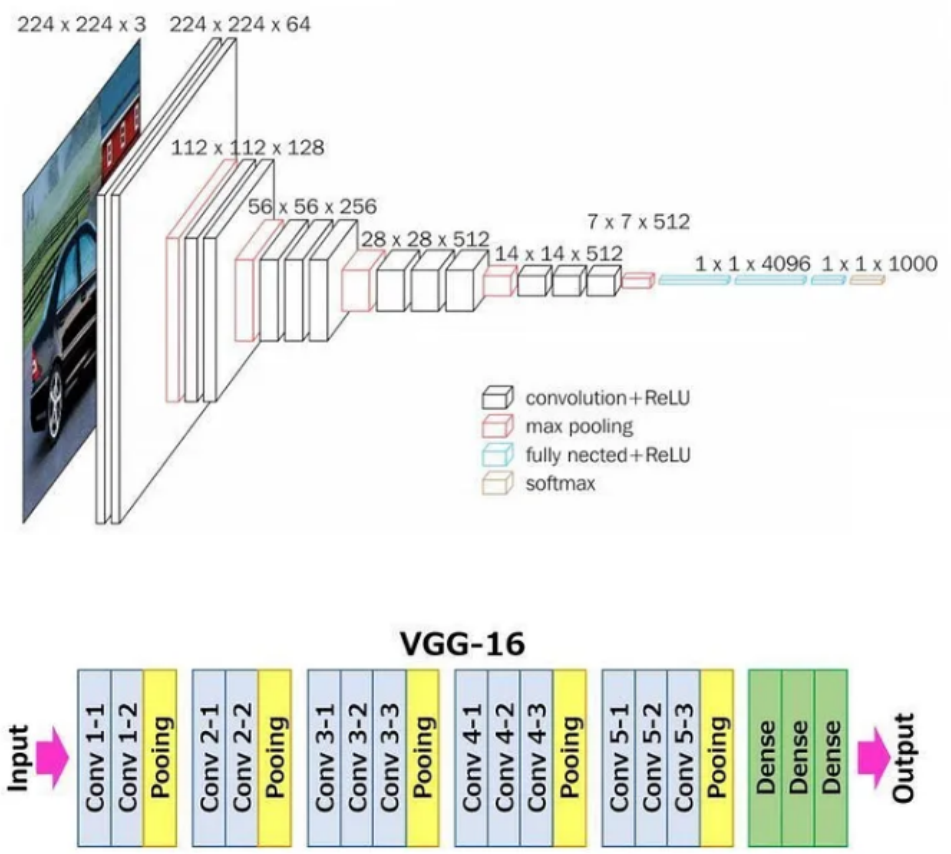

Figura 3.19. Arquitetura da VGG-16, com 16 camadas com pesos treináveis, camadas de *pooling* e camadas densas no final (G, 2021).

A VGG-16 é caracterizada por sua simplicidade e arquitetura uniforme, onde as camadas convolucionais são principalmente filtros de 3x3 aplicados com um *stride* de 1 e *padding* de 1. Suas camadas de max-pooling são filtros de 2x2 com um *stride* de 2. Esse design direto tem sido um modelo de referência em tarefas de visão computacional e influenciou muitas arquiteturas subsequentes de redes neurais (SIMONYAN, 2014) (G, 2021) (NG, 2024).

Com todos os conceitos utilizados neste trabalho apresentados, pode-se mostrar agora o que foi desenvolvido nele com uso dessas ferramentas.

CAPÍTULO 4

TESTES INICIAIS COM BASE DE DADOS REDUZIDA

Os primeiros testes realizados coma base de dados foram apenas no *dataset* de teste, visto que para utilizar a base de dados inteira seria necessário uma grande quantidade de armazenamento e muito tempo de treinamento.

A partir disso, foram realizados processamentos iniciais das imagens para que esses testes pudessem ser realizados com diferentes arquiteturas de CNNs.

4.1 PRÉ-PROCESSAMENTO INICIAL DAS IMAGENS

A primeira etapa dos testes iniciais foi de realizar o pré-processamento das imagens da base de dados de teste. A Figura 4.1 mostra como o conjunto de teste é organizado.

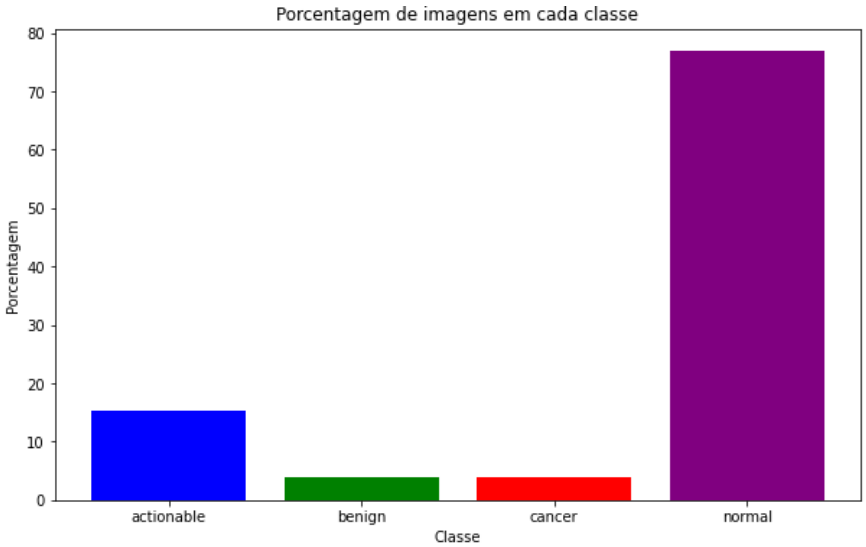

Tabela quantidade-porcentagem:

	Quantidade	Porcentagem
actionable	240	15.374760
benign	61	3.907751
cancer	60	3.843690
normal	1200	76.873799

Figura 4.1. Estrutura do conjunto de teste do *DukeDBT*, nos quais os testes iniciais foram realizados.

Para que fosse possível utilizar as imagens armazenadas nos arquivos .dcm, foi necessário primeiro extraí-las deles. Como mencionado anteriormente, o *dataset* fornece não só os arquivos .dcm, mas também arquivos .csv para que seja possível identificar o que cada um dele representa.

Ao instalá-los, eles são colocados em uma pasta que os separa por ID de paciente (*PatientID*), sem especificar qual o diagnóstico ou qual tipo de vista está sendo representada. Cada pasta de paciente armazena outra única pasta, a qual tem até 4 outras pastas dentro com os arquivos .dcm, que são os diferentes ângulos do exame, mas que também não são especificados explicitamente. Todos os arquivos DICOM têm o mesmo nome, "1-1.dcm".

Como exemplo, tem-se a path de um único arquivo .dcm do conjunto de teste - ".../test/manifest-1617905855234/Breast-Cancer-Screening-DBT/DBT-P00004/01-01-2000-DBT-S03764-MAMMO screening digital bilateral-73497/11920.000000-70893/1-1.dcm".

O arquivo .csv de *paths* disponibiliza o local em que os arquivos serão organizados ao serem baixados, visto que a base de dados possui uma forma específica de fornecer os arquivos .dcm. O arquivo .csv de diagnósticos possibilita identificar a qual classe cada arquivo DICOM pertence. Ambos fornecem também cada vista dos estudos.

As figuras 4.2 e 4.3 mostram os *headers* de cada um desses arquivos, que representam suas estruturas.

	PatientID	StudyUID	View	Normal	Actionable	Benign	Cancer
0	DBT-P00882	DBT-S04273	lcc	0	0	0	1
1	DBT-P00882	DBT-S04273	lmlo	0	0	0	1
2	DBT-P01803	DBT-S04833	rcc	0	0	0	1
3	DBT-P01803	DBT-S04833	rmlo	0	0	0	1
4	DBT-P01183	DBT-S04722	rcc	0	0	0	1

Figura 4.2. *Header* do arquivo .csv utilizado para relacionar o ID do paciente com seu diagnóstico em codificaão *one-hot*.

	PatientID	StudyUID	View	descriptive_path	classic_path
0	DBT-P00036	DBT-S03354	lcc	Breast-Cancer-Screening-DBT/DBT-P00036/01-01-2...	Breast-Cancer-Screening-DBT/DBT-P00036/1.2.826...
1	DBT-P00036	DBT-S03354	lmlo	Breast-Cancer-Screening-DBT/DBT-P00036/01-01-2...	Breast-Cancer-Screening-DBT/DBT-P00036/1.2.826...
2	DBT-P00036	DBT-S03354	rcc	Breast-Cancer-Screening-DBT/DBT-P00036/01-01-2...	Breast-Cancer-Screening-DBT/DBT-P00036/1.2.826...
3	DBT-P00036	DBT-S03354	rmlo	Breast-Cancer-Screening-DBT/DBT-P00036/01-01-2...	Breast-Cancer-Screening-DBT/DBT-P00036/1.2.826...
4	DBT-P00087	DBT-S04170	lcc	Breast-Cancer-Screening-DBT/DBT-P00087/01-01-2...	Breast-Cancer-Screening-DBT/DBT-P00087/1.2.826...

Figura 4.3. *Header* do arquivo .csv utilizado para relacionar o ID do paciente com o local em que o arquivo foi instalado.

A diferença entre *descriptive path* e *classic path* é apenas entre os nomes das *paths* em que os arquivos são instalados, que é uma opção escolhida ao baixá-los. Neste trabalho a primeira

foi utilizada.

Para extrair as imagens dos arquivos .dcm, ambos os .csv foram utilizados junto com um código desenvolvido em *python* para convertê-las em .png, a fim de reduzir o espaço de armazenamento necessário do *dataset* e possibilitar uma visualização mais fácil das figuras. Além disso, pastas para cada diagnóstico também foram criadas para separar cada imagem por classe. É importante ressaltar que para essa conversão para .png, o mesmo canal foi repetido três vezes para os canais RGB.

O caminho de cada arquivo é relacionado sequencialmente junto com o *PatientID* e sua respectiva vista para cada .dcm. Com o local do .dcm definido, extrai-se a informação contida apenas no primeiro canal (i.e. *slice*) da imagem contida nele e são normalizados os valores de seus *pixels* de 0 até 255, visto que originalmente possuíam valores muito altos.

Para a normalização, primeiro toda a matriz de valores da imagem contida no .dcm é dividida pelo valor máximo, o que faz com que maior *pixel* seja igual a 1 e os valores intermediários fiquem entre 0 e 1. Posteriormente, ela é multiplicada por 255, para que todos os *pixels* fiquem entre 0 e 255. Esse processo é mostrado pelas equações 4.1 e 4.2.

$$pixelArray = \frac{pixelArray}{\max(pixelArray)} \tag{4.1}$$

$$pixelArray = pixelArray \times 255 \tag{4.2}$$

Depois da normalização da imagem, ela é convertida para .png e renomeada a partir do *PatientID* e sua vista e é salva na pasta de seu diagnóstico, informações obtidas a partir dos .csvs. A Figura 4.4 representa como a estrutura de uma das pastas fica após esse processo.

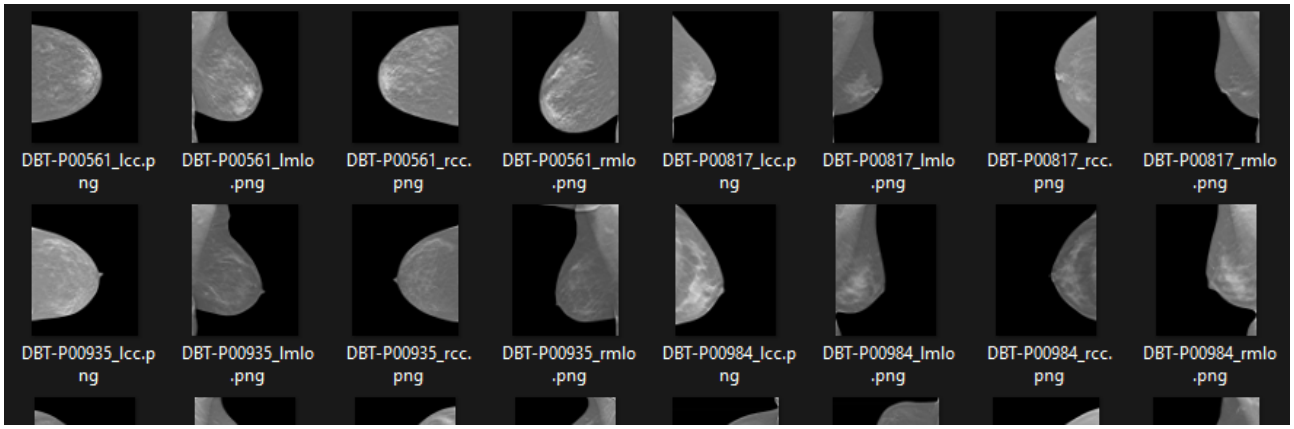

Figura 4.4. Pasta das mamografias com diagnóstico actionable do dataset de teste após extração e conversão para .png e renomeação a partir do *PatientID* e vista.

Com as imagens efetivamente separadas por classes dos conjuntos de treino, teste e validação, foi possível realizar testes iniciais para ver como CNNs simples responderiam a elas durante o treinamento. É importante ressaltar que inicialmente os testes tinham sido feitos ainda com a classe *actionable*.

4.2 ARQUITETURAS INICIAIS DE CNNS

As imagens dos arquivos DICOM que foram extraídas possuem resoluções bem altas, as quais possuem dimensão de (1996x2457x3) ou (1890x2457x3). Para estes testes iniciais, elas foram reduzidas consideravelmente, para (256x256x3), a fim de verificar se as redes seriam capazes de identificar as diferenças entre as 4 classes. Esse redimensionamento pode ser verificado na Figura 4.5.

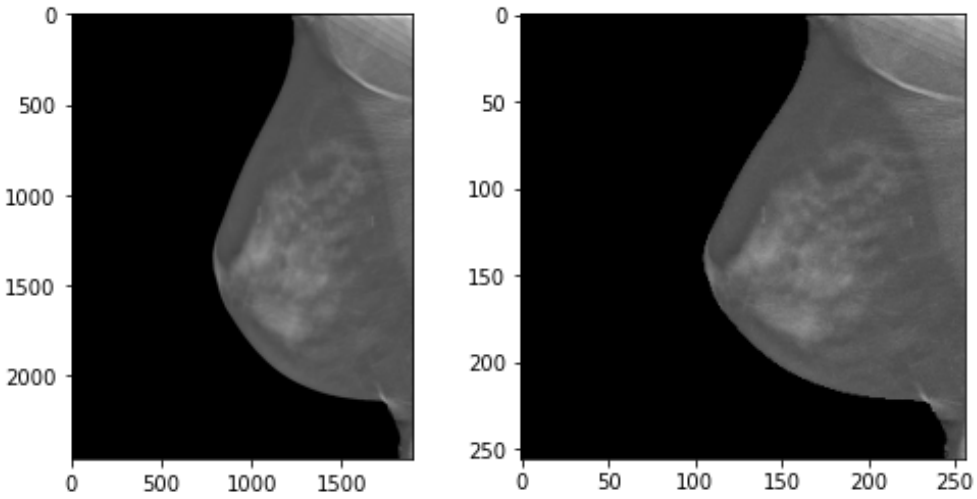

Figura 4.5. Redimensionamento inicial das imagens do *dataset* para (256x256x3) antes de serem alimentadas na rede. Apesar da resolução ter diminiuído, ainda são notáveis os detalhes da figura.

4.2.1 CNN simples para testes iniciais

Com as imagens prontas, elas foram divididas em conjuntos de treino, teste e validação, com uma divisão de, respectivamente, 70%, 15% e 15% e *batch size* de 32. O primeiro teste foi realizado com uma rede simples criada do zero, a qual tem arquitetura mostrada na Figura 4.6. Todos os treinamentos deste trabalho foram realizados com uso de *tensorflow* e *python*.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
batch_normalization (BatchNormalization)	(None, 254, 254, 32)	128
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 125, 125, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
flatten (Flatten)	(None, 246016)	0
dense (Dense)	(None, 128)	31,490,176
dense_1 (Dense)	(None, 4)	516

Figura 4.6. Arquitetura simples utilizada em testes iniciais no *dataset* de teste, composta de camadas convolucionais, *pooling*, normalização e densas.

O treinamento desta rede foi realizado utilizando a técnica de regularização de *early stopping* para interromper o treinamento antes que a perda de validação comece a aumentar significativamente, a fim de evitar um possível *overfitting*.

Para este caso, aplicou-se o *early stopping* com uma paciência de 6 épocas, ou seja, o treinamento é interrompido se a perda de validação não apresentar melhora após 6 épocas consecutivas. Se isso ocorrer, o treinamento é cessado e os pesos correspondentes à menor perda de validação são recuperados e salvos no modelo.

Antes das imagens serem alimentadas na rede, os valores dos *pixels* das bases de treino, validação e teste são normalizadas para terem valores entre 0 e 1, ou seja, elas têm todos os seus valores divididos por 255. Essa será a normalização utilizada para todas as redes, exceto para a ResNet-50, a qual necessita de um procedimento específico.

Por fim, foi utilizado o otimizador Adam com taxa de aprendizado igual a 0,0001 e função de perda de entropia cruzada categórica. A Figura 4.7 mostra os gráficos da acurácia e de perda do treinamento e validação.

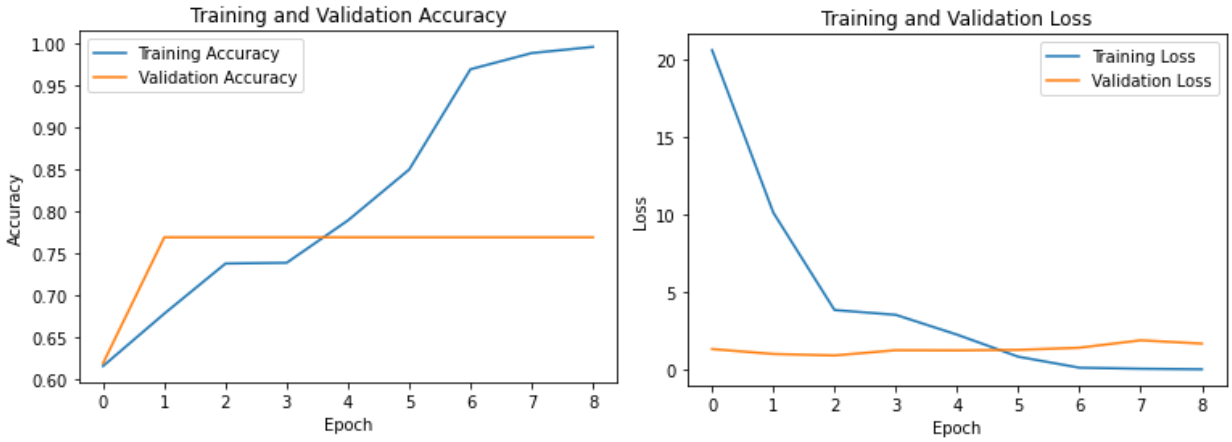

Figura 4.7. Gráficos gerados do treinamento e validação da rede simples desenvolvida com *early stopping* de paciência de 6 épocas. Percebe-se do gráfico de acurácia que bem cedo no treinamento a rede já seguia para um *overfitting*, representado pela acurácia de treinamento se aproximando de 100% e a de validação estagnada.

A rede teve acurácia de 76,6% e perda de 0,9164 nas imagens de teste. Inicialmente parecem ser bons resultados, mas na realidade ela jogou todas as suas predições na classe majoritária, de imagens de diagnóstico normal. Isso pode ser observado na matriz de confusão representada na Figura 4.8.

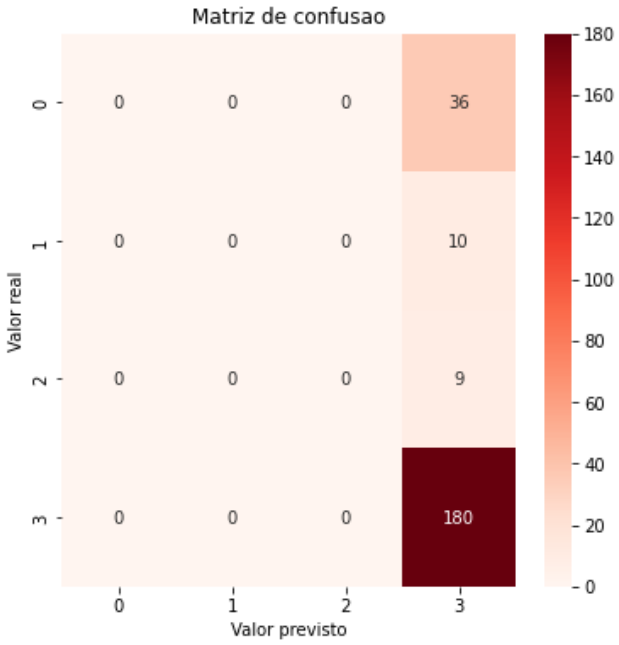

Figura 4.8. Matriz de confusão gerada a partir da aplicação do modelo no conjunto de teste. Os valores de 0 a 3 representam, respectivamente, as classes *actionable*, *benign*, *cancer* e normal.

Na Figura 4.8, os valores de 0 até 3 nos eixos verticais e horizontais representam, respectivamente, as classes *actionable*, *benign*, *cancer* e normal, enquanto que os valores dentro dos quadrados mostram a quantidade de predições realizadas em determinada classe. Por exemplo, o valor 36 no canto superior direito representa que a rede classificou 36 casos *actionable* (valor real, representado pelo eixo vertical) como normais (valor previsto, representado pelo eixo horizontal).

Após os testes com a rede original, tentou-se aplicar *trasnfer learning* com uso da VGG-16.

4.2.2 *Transfer learning* com a VGG-16

Para o uso da VGG-16, seus pesos foram congelados e após a rede original foi adicionada uma operação de *flatten* para que as informações das saídas das redes convolucionais pudessem ser alimentadas a uma camada densa de 256 neurônios, também adicional. Por fim, uma camada de *softmax* de 4 classes foi colocada no fim para a classifição das imagens de entrada.

As mesmas configurações para a regularização, otimização e perda anteriores foram utilizadas. O resultado do treinamento e validação é representado na Figura 4.9. O teste teve acurácia de 69,39% e perda de 0,8604.

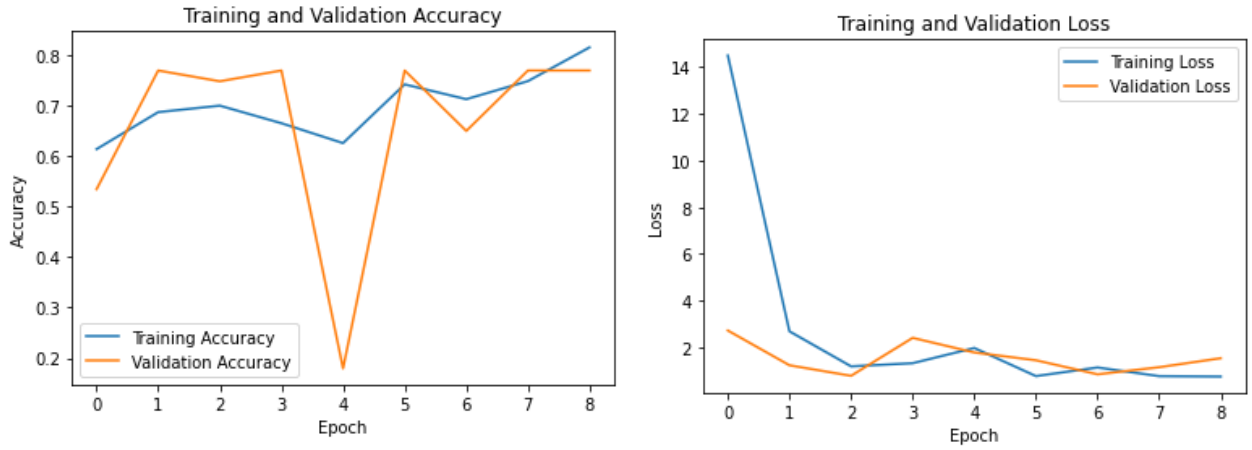

Figura 4.9. Gráficos de treinamento e validação dos testes iniciais com a VGG-16.

A partir dos gráficos da Figura 5.9, percebe-se que a rede teve dificuldade para chegar em um *overfitting* quando comparada à CNN anterior, visto que sua acurácia flutuou na casa do 80% no treinamento. Apesar disso, durante o teste, ela foi capaz de classificar algumas das imagens como casos *actionable*, o que pode ser visualizado na Figura 4.10.

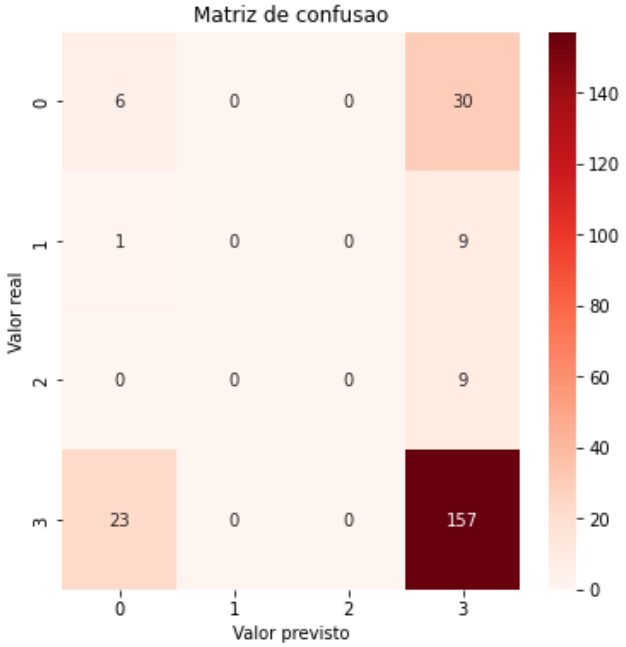

Figura 4.10. Matriz de confusão dos testes inicias com a VGG-16. Nela é notável que classificou alguns dos casos para a classe *actionable*

Por fim, para os testes iniciais, tentou-se também *transfer learning* com uso da DenseNet-121.

4.2.3 *Transfer learning* com a DenseNet-121

Para o *transfer learning* com a DenseNet-121, a mesma configuração para as camadas após a rede da VGG-16 foram utilizadas, além de também as mesmas configuração de regularização, otimização e de perda e seus pesos foram congelados. Os gráficos resultantes do treinamento e validação podem ser vistos na Figura 4.11. O teste teve acurácia de 76,6% e perda de 0,98.

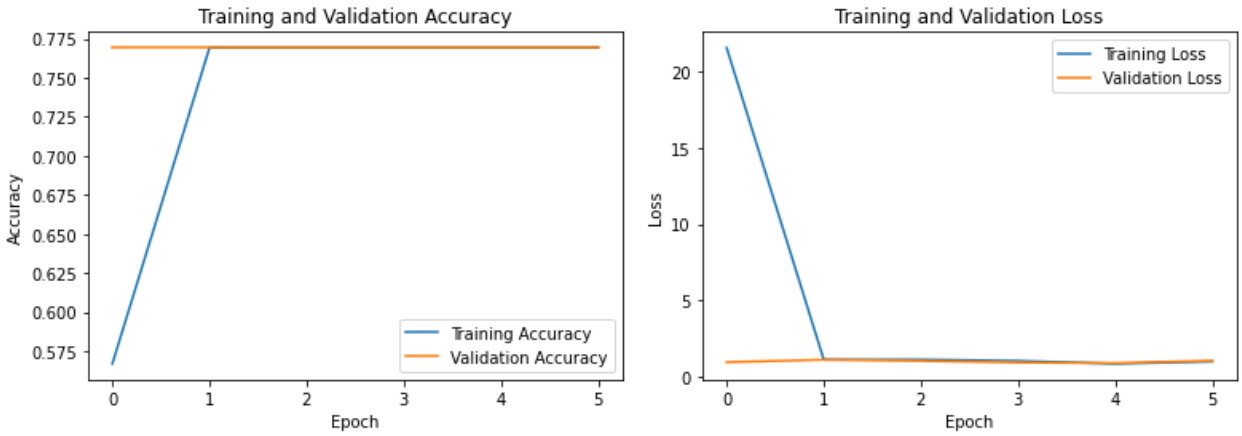

Figura 4.11. Gráficos de treinamento e validação dos testes iniciais com DenseNet-121. Percebe-se como rapidamente a rede chegou em um valor de acurácia e perda que tiveram pouca variação nas próximas épocas.

A partir dos gráficos da Figura 4.11, percebe-se que a rede teve um pico de acurácia e um vale para a perda, nos quais não foi capaz de melhorar. O resultado do teste foi muito próximo com o da CNN simples, que pode ser observado na matriz de confusão da Figura 4.12.

A partir dos resultados das três redes, concluiu-se que seria possível ter melhores resultados com uma base de dados binária, visto que a quantidade de imagens é desbalanceada com muitos casos de classificação normal.

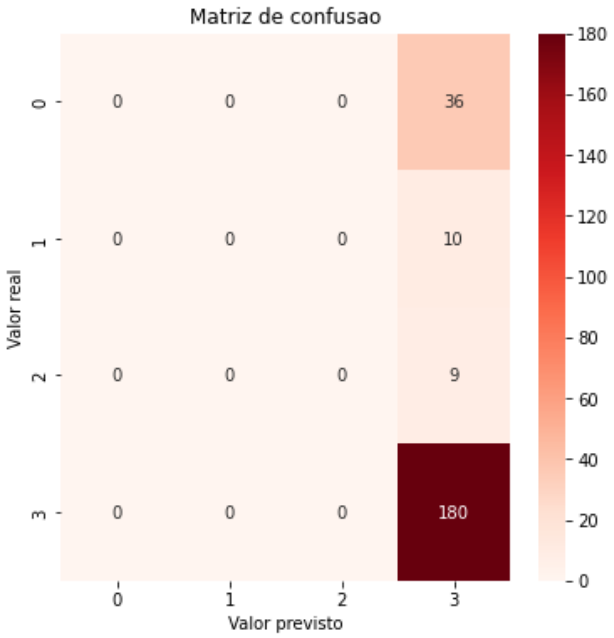

Figura 4.12. Matriz de confusão dos testes inicias com a DenseNet-121. Percebe-se que teve os mesmos resultados que a CNN desenvolvida do zero.

4.3 TESTES INICIAIS COM BASE DE DADOS BINÁRIA

Nestes testes, foram utilizadas as mesmas configurações das três redes desenvolvidas anteriormente, mantendo as mesmas dimensões para as imagens de entrada. No entanto, a saída e a perda foram ajustadas para uma classificação binária e com *early stopping* com paciência de 10, aplicada a um *dataset* que foi dividido em apenas duas classes; normal e *other* (que inclui as classes *actionable, benign* e *cancer*) para classificação binária.

4.3.1 CNN simples para testes binários iniciais

A rede simples desenvolvida chegou em um *overfitting* durante o treinamento bem rápido, visto que a partir da 7ª época já havia chegado em uma acurácia bem próxima de 100%, porém com uma perda de validação cada vez maior. Esses resultados podem ser vistos na Figura 4.13.

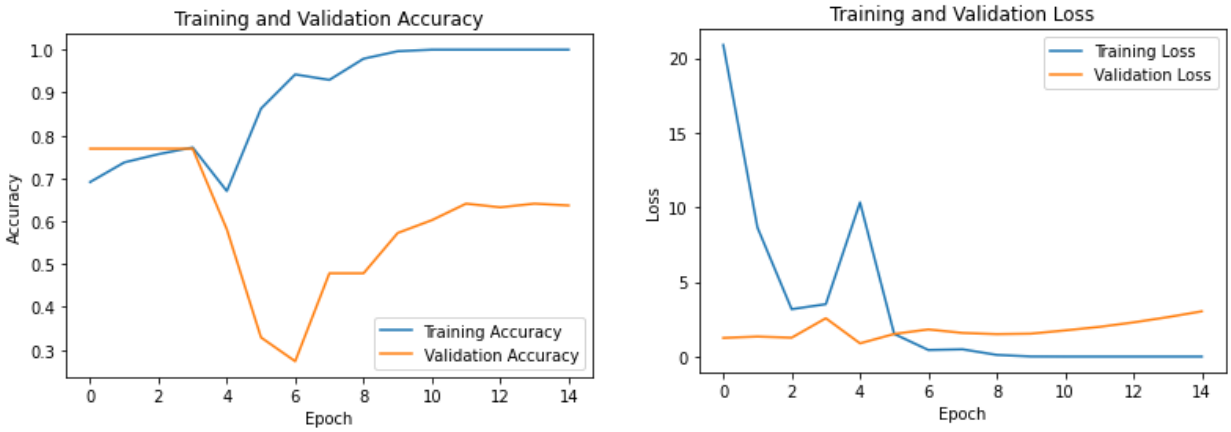

Figura 4.13. Treinamento e validação para a classificação binária com a CNN simples. Percebe-se um caso de *overfitting* a partir da 7ª época, visto que o treinamento chega em acurácia de próxima de 100% mas sem resultados bons na validação.

No teste, teve-se uma acurácia de 60% com perda de 0,7847. Apesar disso, pelos resultados que podem ser vistos na matriz de confusão da Figura 4.14, percebe-se que a rede foi capaz de começar a separar os casos em classes diferentes.

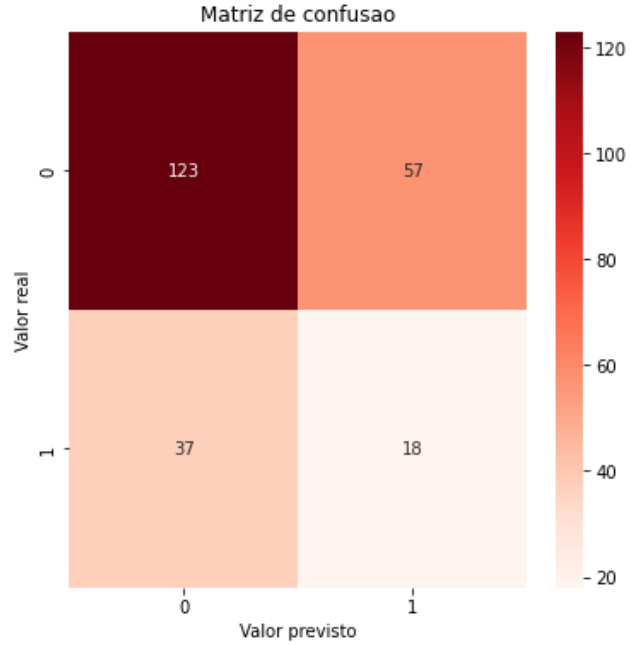

Figura 4.14. Matriz de confusão para a classificação binária inicial com a CNN simples. É notável que com classificação binária a rede começa a separar alguns dos casos de teste, em que a classe 0 representa os casos normais e 1 representa os casos *other*.

Após os testes com a CNN simples, tentou-se novamente um treinamento com *transfer learning* com a DenseNet-121 e a VGG-16.

4.3.2 *Transfer learning* com DenseNet-121 e VGG-16

Tanto a DenseNet-121 quanto a VGG-16 apresentaram dificuldades durante o treinamento e validação, alcançando uma acurácia de 75% na primeira fase e estagnando na segunda, o que pode ser observado nos gráficos da Figura 4.15.

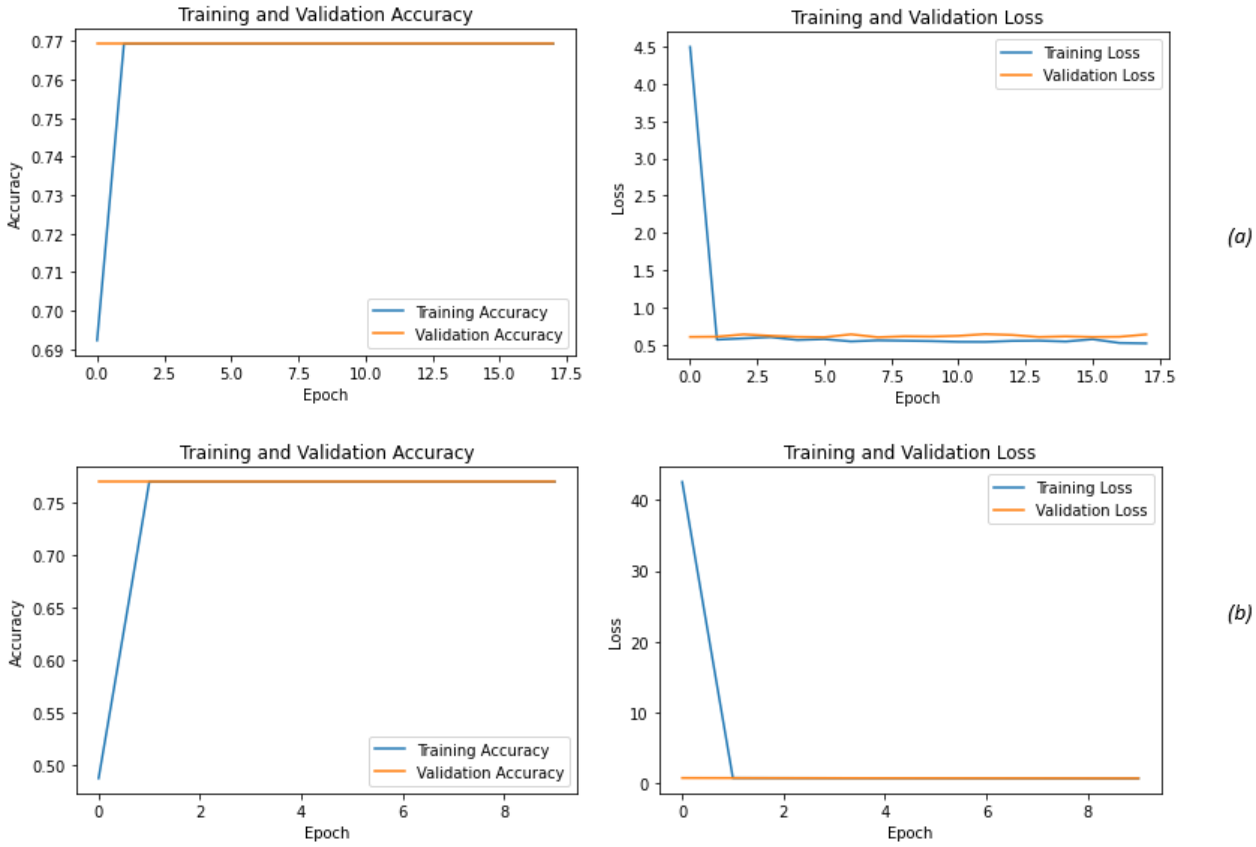

Figura 4.15. Gráficos de trainamento e validação na classificação binária inicial com a VGG-16 (a) e DenseNet-121 (b). É perceptível como ambas chegaram em um ponto de estagnação tanto no treinamento e validação.

Durante a fase de teste, ambas as rede tiveram acurácia de 76,6%, classificando todas as imagens como a classe majoritária durante o treinamento (normal), o que pode ser visualizado nas matrizes de confusão da Figura 4.16.

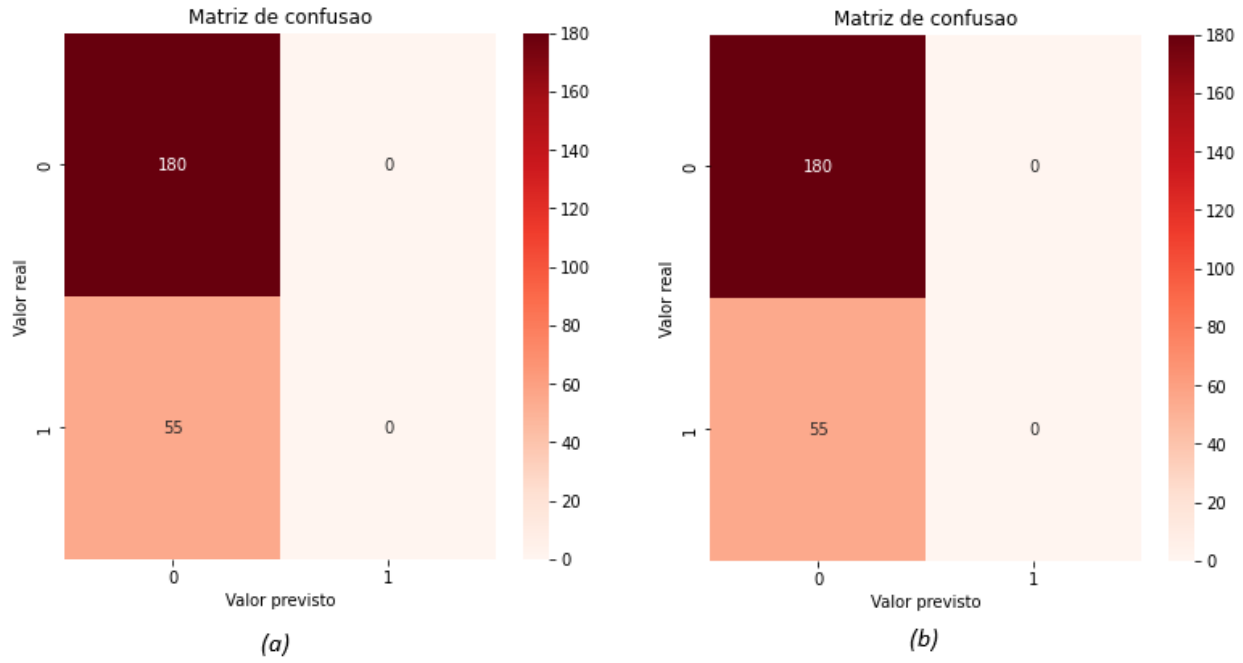

Figura 4.16. Matrizes de confusão na classificação binária inicial com a VGG-16 (a) e DenseNet-121 (b). Nota-se como ambas tiveram os mesmos resultados, classificando todas as imagens de teste como a classe majoritária de treinamento (normal).

A partir destes testes, supôs-se que seria possível tem um melhor desempenho a partir do uso de imagens com resoluções maiores, visto que provavelmente ocorre perda de informação com o redimensionamento realizado. Além disso, os próximos testes realizados foram feitos com uma base de dados balanceada para analisar como isso afetaria o treinamento da rede.

4.3.3 Testes iniciais com base de dados balanceada

Nestes testes foi feita uma divisão da base de dados para que ela ficasse o mais balanceada possível, separando cada classe em valores próximos um dos outros. A divisão implementada pode ser visualizada na Figura 4.17.

Além do balanceamento, também foi definido um novo valor para o redimensionamento das imagens de entrada da rede. Como mencionado anteriormente, há dois possíveis tamanhos para as figuras: (1996x2457x3) e (1890x2457x3). O redimensionamento foi baseado na menor resolução, aplicando uma redução de escala de 4x, resultando em imagens de tamanho (472x614x3) para a entrada da rede.

O testes realizado com essas imagens foram feitos a partir de *transfer learning* com a DenseNet-121 e a VGG-16, com uma divisão de 70%/15%/15% para os conjuntos treino, vali-

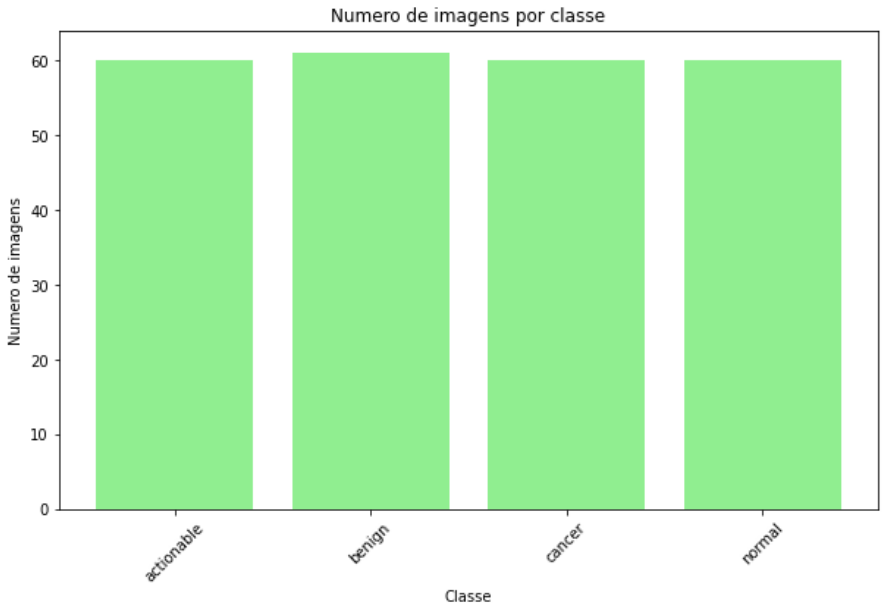

Figura 4.17. Gráfico da base de dados balanceada dos testes iniciais, na qual a classe *benign* possui 61 imagens e as restantes possuem 60.

dação e teste.

Ambas as redes tiveram a mesma configuração, com uso de um *batch size* de 8, uso do otimizador Adam com taxa de aprendizagem padrão de 0,001 e perda de entropia cruzada categórica. Seus pesos foram mantidos congelados e adicionou-se uma operação de *flatten* após ambas as redes, além de uma camada densa de 256 neurônios seguida de uma *softmax* para 4 classes. Elas foram treinadas dessa vez sem *early stopping* por 50 épocas para ver como um treinamento longo afetaria a perda na validação.

4.3.4 *Transfer learning* com a DenseNet-121

Com as configurações mencionadas, a DenseNet-121 não foi capaz de ter um treinamento nem validação efetivos, com acurácia de treinamento flutuante e de validação estagnada. Além disso, ambas as perdas rapidamente chegaram em um valor sem variar mais, o que pode ser visualizado na Figura 4.18.

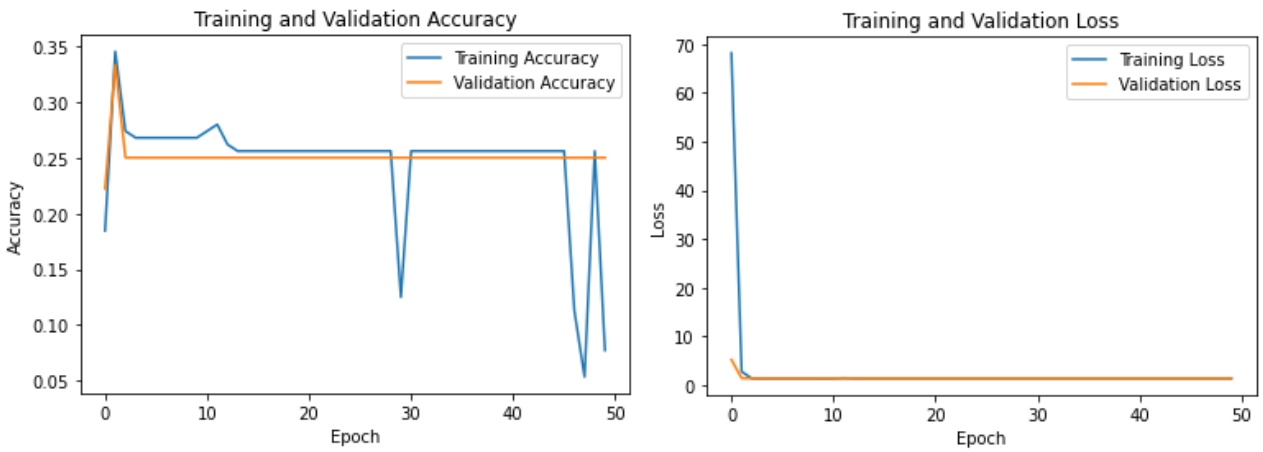

Figura 4.18. Treinamento e validação da DenseNet-121 com classes balanceadas. Percebe-se como a acurácia de treinamento flutuou bastante, enquanto que a acurácia de validação e ambas as perdas estagnaram rapidamente

Para os testes, a rede fez todas as suas classificações na classe *benign*, com acurácia de 27,03%, que é representado na matriz de confusão da Figura 4.19.

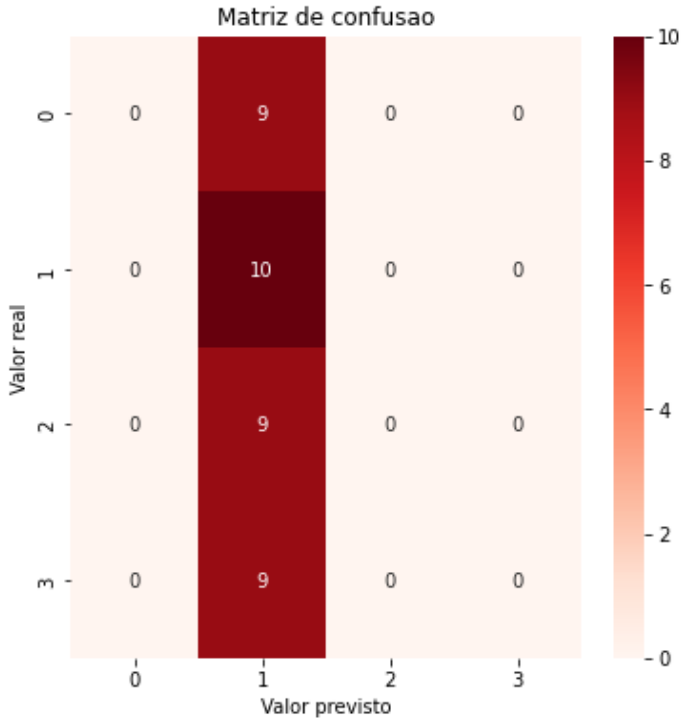

Figura 4.19. Matriz de confusão da base balanceada com DenseNet-121. O resultado revela que a rede classificou todas as imagens como pertencentes à classe *actionable*. O treinamento insatisfatório sugere que a rede não conseguiu aprender as características distintivas das classes, mas simplesmente fez predições aleatórias.

4.3.5 *Transfer learning* com a VGG-16

Comparado com os resultados da DenseNet-121, o treinamento da VGG-16 foi um pouco melhor, visto que a rede chegou em um *overfitting* após 21 épocas, mas continuou com resultados ruins de validação, o que é mostrado nos gráficos da Figura 4.20.

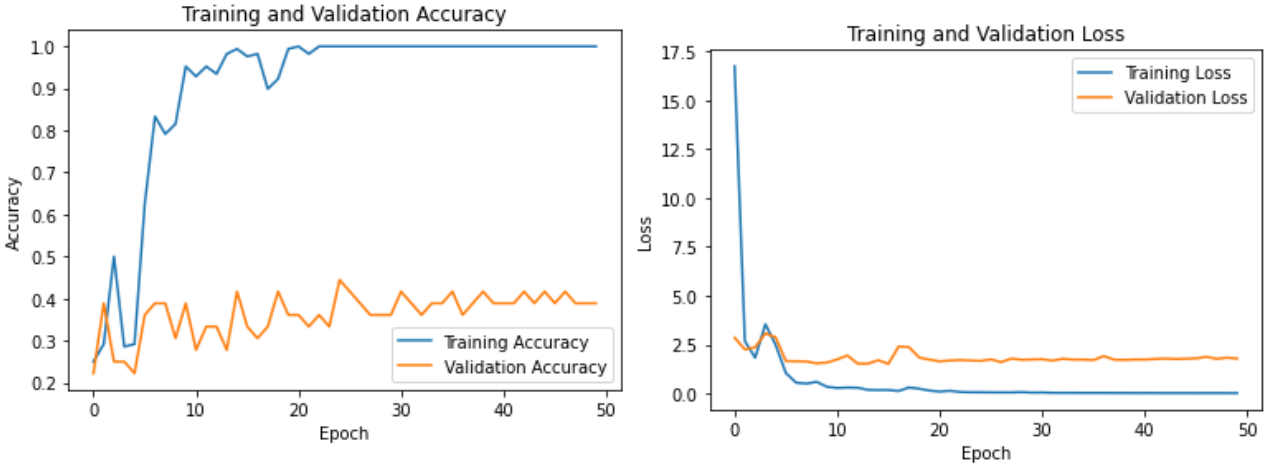

Figura 4.20. Treinamento e validação da VGG-16 com classes balanceadas. É notável como no treinamento a rede atinge um *overfitting*, visto que a acurácia chega em 100% sem ter melhoras em sua validação.

Já no teste, a rede foi capaz de classificar as imagens em classes diferentes, o que pode ser visto na matriz de confusão da Figura 4.21, mas a CNN ainda não teve resultados bons, com acurácia de 27,03%.

Visto que os resultados seguiram insatisfatórios, os próximos treinamentos realizados foram com uso da técnica de regularização de *data augmentation*, que consiste em aplicar operações como espelhamentos verticais e horizontais nas imagens de treinamento, para que a rede possa analisar "novas"figuras e ter uma base de treino maior.

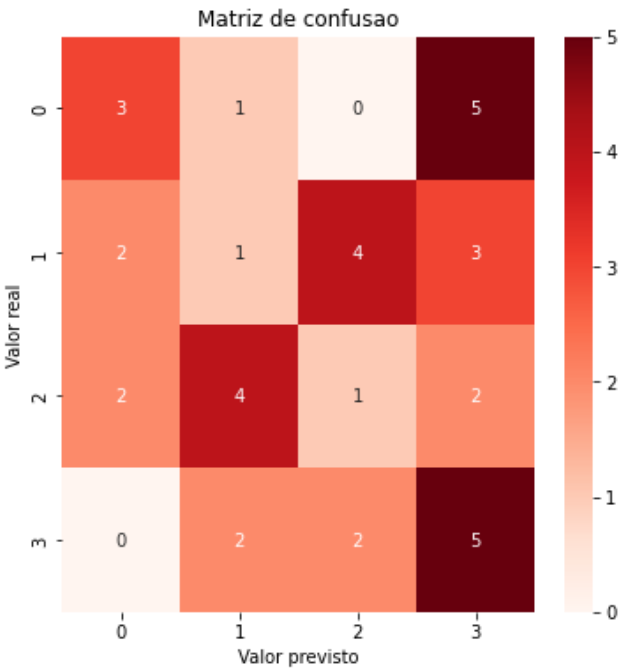

Figura 4.21. Matriz de confusão da base balanceada com VGG-16. Percebe-se que a rede foi capaz de classificar as imagens em todas as classes, mas não de forma efetiva.

4.4 TESTES INICIAIS COM BASE DE DADOS BALANCEADA E APLICAÇÃO DE *DATA AUGMENTATION*

Diversos testes com diferentes tipos de *data augmentation* foram realizados. O que resultou em resultados melhores foi com uso da base de dados balanceada com aplicação das operações de espelhamento vertical, horizontal e ambos simultaneamente. Além disso, também foram alterados os valores dos *pixels* das imagens de forma aleatória entre valores de -50 até 50 (que representam mudanças no brilho das imagens), aplicados ajustes de contraste aleatórios e, por fim, também foi adicionado ruído.

Essas operações foram aplicadas com uso da biblioteca OpenCV (CV2) do *python*, o que permitiu um aumento do conjunto de treinamento de 168 imagens para 1176. É importante ressaltar que essas operações foram aplicadas separadamente nas imagens originais e apenas no conjunto de treinamento, visto que o conjunto de validação e o de teste devem ter a mesma distribuição.

Além disso, a grande maioria dos próximos testes foram feito com uso apenas da DenseNet-121, visto que é o modelo utilizado no *paper* do *dataset* (BUDA, 2021).

4.4.1 *Transfer learning* com a DenseNet-121

O *transfer learning* foi feito com congelamento geral da rede e adição de uma operação de *flatten* seguida por uma camada densa de 128 neurônios. Por fim, foi colocada uma camada *softmax* para a classificação dos 4 possíveis diagnósticos. A rede foi treinada por 50 épocas com uso do otimizador Adam com taxa de aprendizagem 0,0001 e perda de entropia cruzada categórica. Além disso, utilizou-se um *batch size* de 32.

A rede aparenta atingir um *overfitting* após a 13ª época, quando atingiu acurácia de 100%, mas não conseguiu ter melhorias na acurácia e perda de validação, as quais flutuaram bastante. Os resultados podem ser vistos na Figura 4.22.

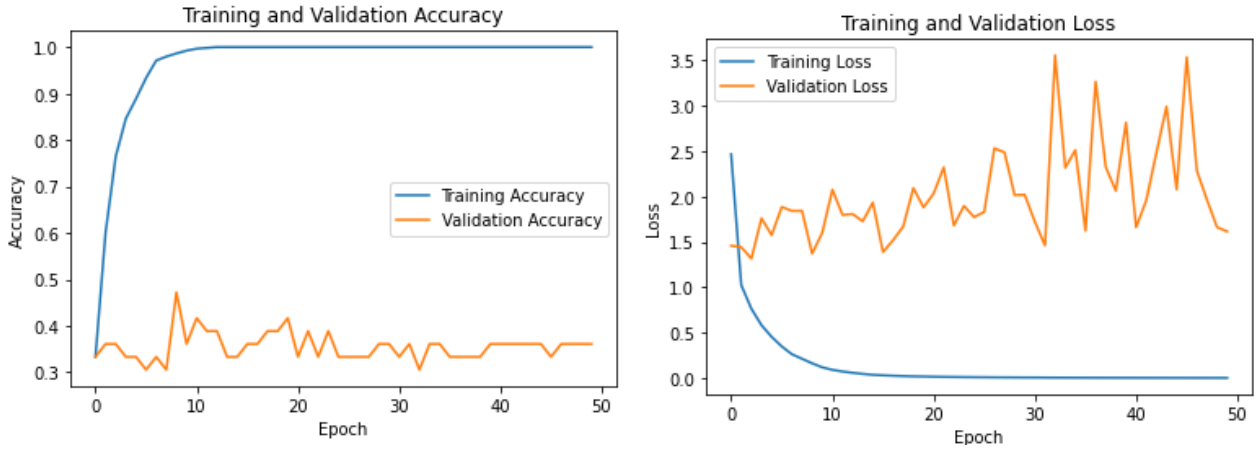

Figura 4.22. Treinamento e validação da DenseNet-121 com classes balanceadas e *transfer learning*. É perceptível como a rede chegou em um *overfitting* mas não conseguiu aprender as características da base de dados, visto que a acurácia e perda na validação flutuaram constantemente.

Assim como anteriormente, os testes seguiram insatisfatórios, com uma acurácia de apenas 16,22% e perda de 3,7664. A matriz de confusão gerada para os testes pode ser visualizada na Figura 4.23.

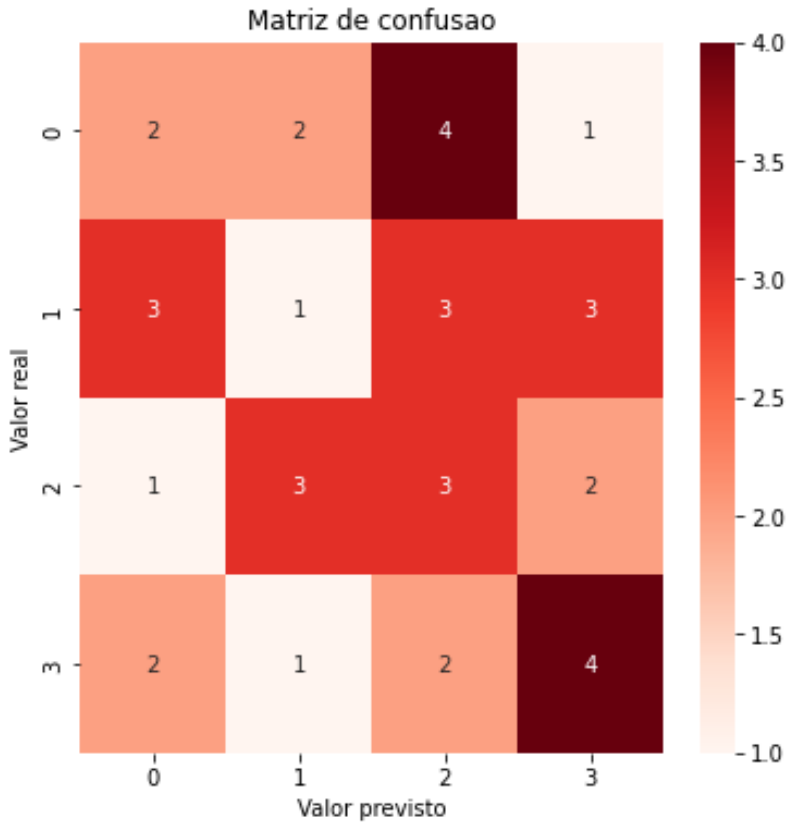

Figura 4.23. Matriz de confusão da DenseNet-121 com classes balanceadas e *transfer learning*. A partir dela, é notável como a rede parece chutar a classificação das imagens aleatoriamente dentre as possíveis classes.

Visto que os resultados dos testes seguiram insatisfatórios, as próximas etapas do projeto foram realizadas com a base de dados completa, a fim de analisar se a adição de uma maior quantidade de imagens melhoraria o desempenho da rede.

CAPÍTULO 5

TESTES COM A BASE DE DADOS COMPLETA

Devido aos resultados insatisfatórios com uma base de dados reduzida, supôs-se que o motivo por trás da falta da capacidade da rede de aprender as características das imagens da base de dados era devido à baixa quantidade delas. Com isso, a próxima etapa do projeto foi de treinamento de redes com uso do *dataset* completo.

5.1 PRÉ-PROCESSAMENTO COMPLETO DAS IMAGENS

Os mesmos procedimentos feitos anteriormente para a base de teste foram realizados também para a base de validação e treinamento, o que resultou em bases de validação e treino dividas da forma representada pela Figura 5.1.

Provavelmente devido ao grande desbalanceamento de dados, o *paper* do *dataset* propõe uma separação diferente das imagens, a qual reduz em grande quantidade o número de imagens da classe normal que são utilizadas. Essa divisão é mostrada na Figura 5.2 (BUDA, 2021).

Além disso, o *paper* reduz também a quantidade de imagens das outras classes. Apesar disso, para este trabalho, optou-se pelo uso da quantidade especificada para os casos da classe normal, mas foram utilizadas todas as imagens disponíveis para as restantes, a fim de reduzir um pouco o desbalanceamento e, possivelmente, melhorar o desempenho do treinamento. A separação utilizada é representada na Figura 5.3.

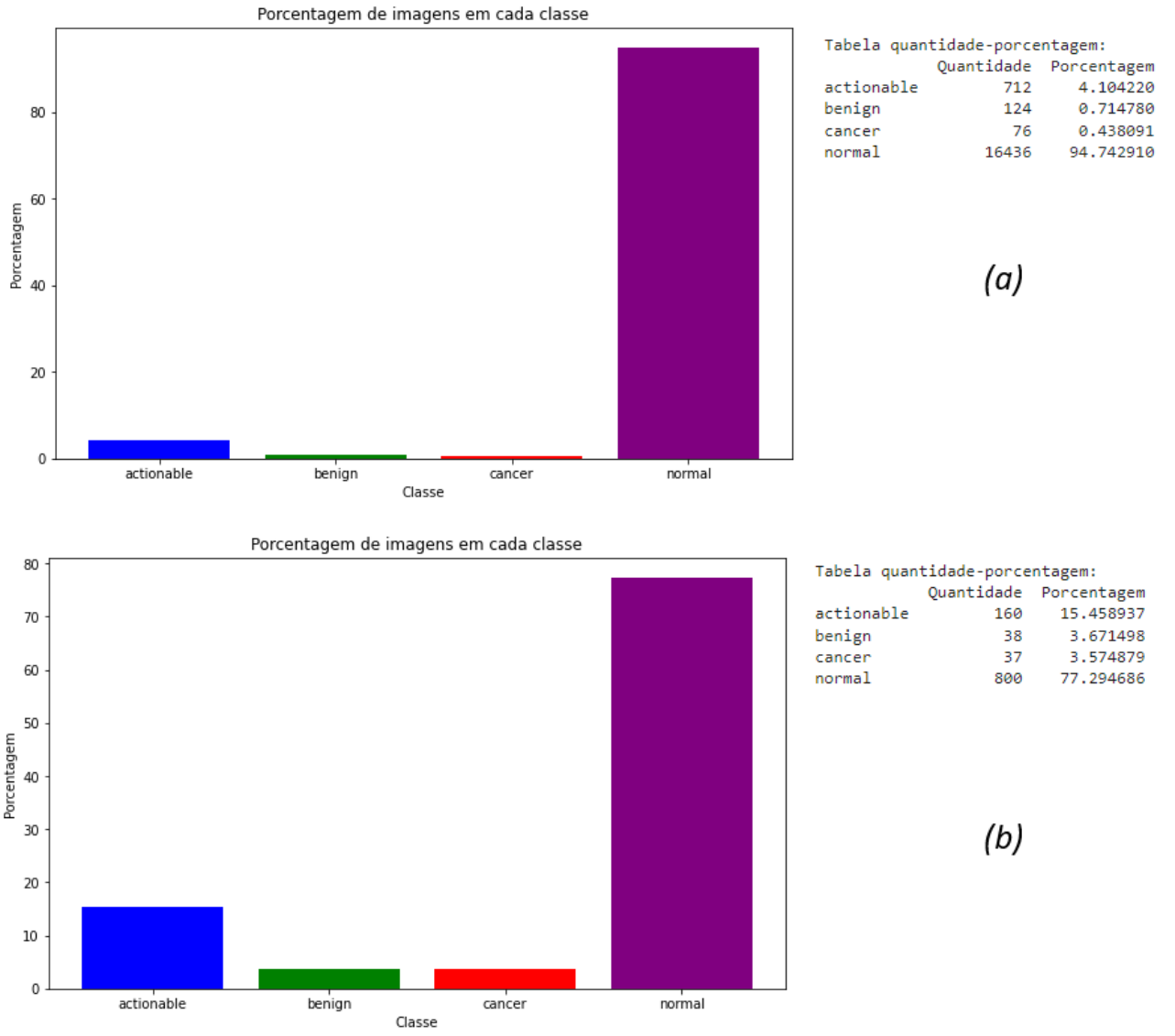

Figura 5.1. Conjunto de treinamento (a) e de validação (b) completos do *dataset*. Percebe-se que, assim como a base de treinamento, tem-se um desbalanceamento alto de dados, em que a maioria das classificações são do diagnóstico normal.

Table 1. Descriptive Statistics of the Data Set Used for Training, Validation, and Testing

Characteristics	No.		
	Training set	Validation set	Test set
Patients			
Total	4362	280	418
Normal group, No. (%)	4109 (94.2)	200 (71.4)	300 (71.8)
Actionable group, No. (%)	178 (4.1)	40 (14.2)	60 (18.9)
Benign group, No. (%)	62 (1.4)	20 (7.1)	30 (7.2)
Cancer group, No. (%)	39 (0.9)	20 (7.1)	30 (7.2)

Figura 5.2. Separação do *dataset* proposta pelo *paper*. Destaca-se como ele reduz em grande quantidade o número de imagens pertencentes à classe normal (BUDA, 2021).

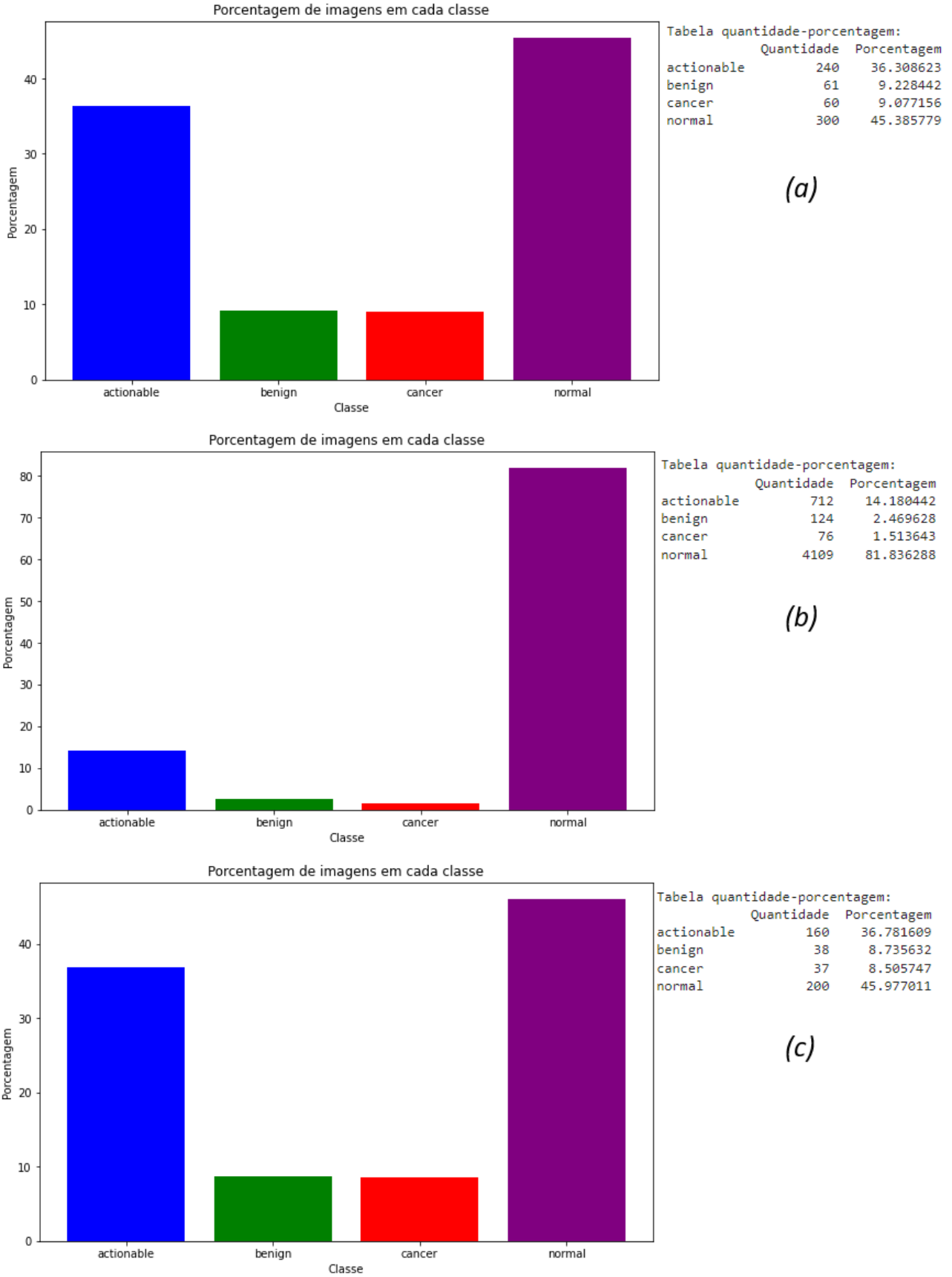

Figura 5.3. Separação do *dataset* proposta pelo trabalho, a qual teve uma maior quantidade de imagens das classes minoritárias incluídas. Os gráficos representam a base de teste (a), treinamento (b) e validação (c).

5.2 TESTES COM A DENSENET-121 COM USO DA BASE DE DADOS COMPLETA

Como mencionado anteriormente, neste momento, o treinamento da base de dados focou no uso da DenseNet-121. Foram realizados diversos testes com essa rede, com uso de diferentes arquiteturas adicionais após o *transfer learning* e com uso das quatro classes e também classificação binária.

5.2.1 *Transfer learning* com a DenseNet-121 com quatro classes

Com uso das quatro classes da base de dados, diferentes arquiteturas foram adicionadas após o *transfer learning*. Além disso, foi testado o descongelamento gradual das últimas camadas e diversas técnicas de *data augmentation*, mas nenhum desses testes produziu resultados satisfatórios.

A configuração que trouxe os melhores resultados envolveu o uso de *data augmentation* apenas nas classes minoritárias de treinamento, ou seja, *actionable*, *benign* e cancer. Foram aplicadas operações de espelhamento vertical e horizontal, além da adição aleatória de ruído nas imagens. A divisão da base de dados após a aplicação dessas operações pode ser vista na Figura 5.4.

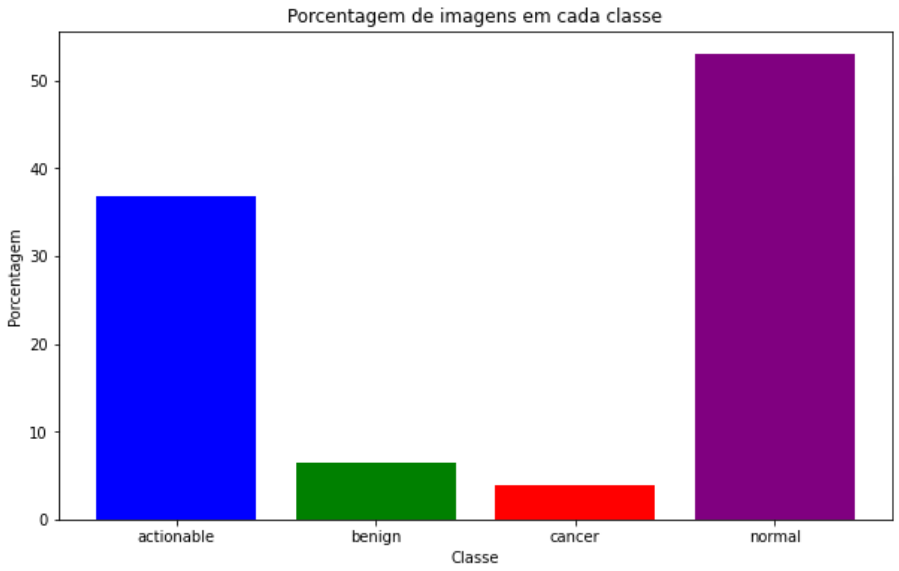

Figura 5.4. Base de dados pós *data augmentation*. Percebe-se um aumento na quantidade das classes minoritáiras, mas o desbalancieamento continua presente, visto que as imagens diangósticas como normais ainda compõem por volta de 50% do *dataset*.

O modelo utilizado para o treinamento foi a partir do uso da *transfer learning* com DenseNet-121 completamente congelada com *batch size* de 16. Após a rede original, foi aplicada uma operação de *flatten* seguida por uma camada densa de 256 neurônios e, por fim, uma *softmax* de quatro classes. O otimizador utilizado foi o Adam, com taxa de aprendizado igual a 0,001. A perda usada foi a entropia cruzada categórica. Além disso, foi utilizado um *early stopping* com paciência de 25 épocas.

Durante o treinamento, a rede aproximou-se de um *overfitting*, mas o *early stopping* impediu que a rede efetivamente chegasse em uma acurácia de 100%. Os resultados do treinamento e validação podem ser vistos na Figura 5.5.

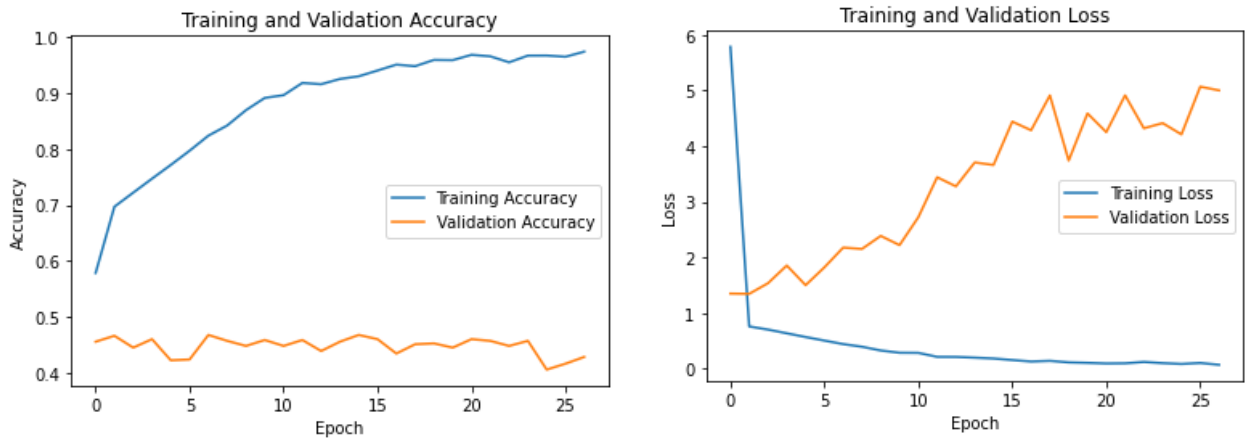

Figura 5.5. Treinamento e validação da DenseNet-121. Percebe-se como desde o começo do treinamento a rede já teve dificuldades de ter bons resultados na validação desde o começo.

Os testes não tiveram resultados bons, com acurácia de 46,67% e perda 1,2255. Os resultados podem ser visualizados na matriz de confsão da Figura 5.6.

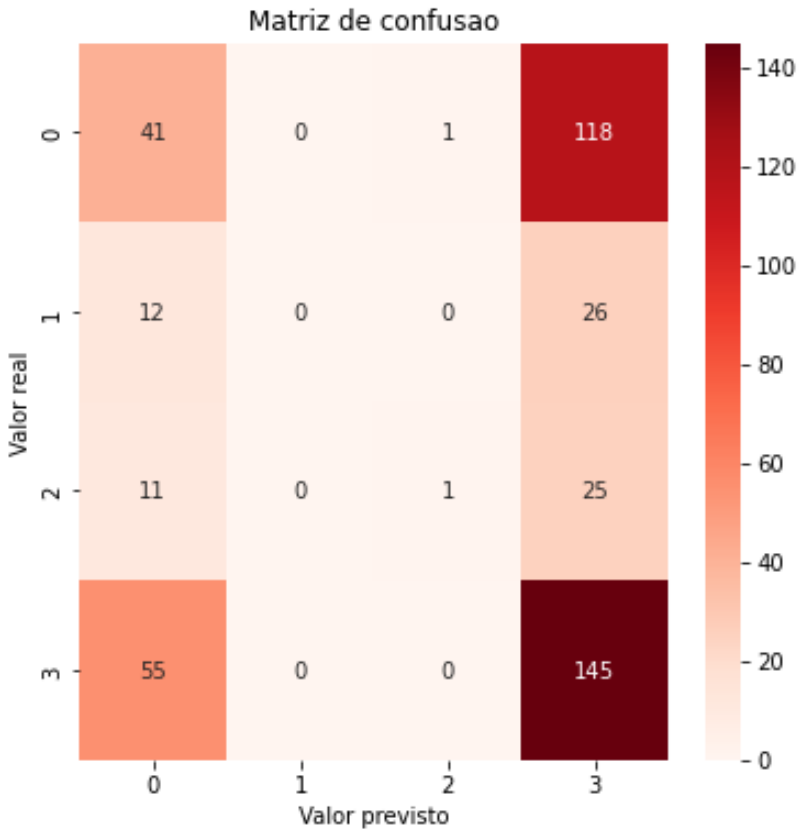

Figura 5.6. Matriz de confusão dos testes com a DenseNet-121. Perecbe-se que a rede colocou algumas das imagens previstas na classe *actionable*, mas a maioria ainda foi classificada como normal.

Assim como anteriormente, os próximos testes realizados foram com uso de uma base dados binária, a fim de verificar como isso afetaria o treinamento da rede.

5.2.2 *Transfer learning* com a DenseNet-121 com classifição binária

Nestes testes, as imagens que resultaram em melhores resultados vieram da base com *data augmentation* aplicado da Figura 5.4. Ao dividir-se as classes de forma binária (*cancer type* e normal), tem-se a configuração mostrada na Figura 5.7.

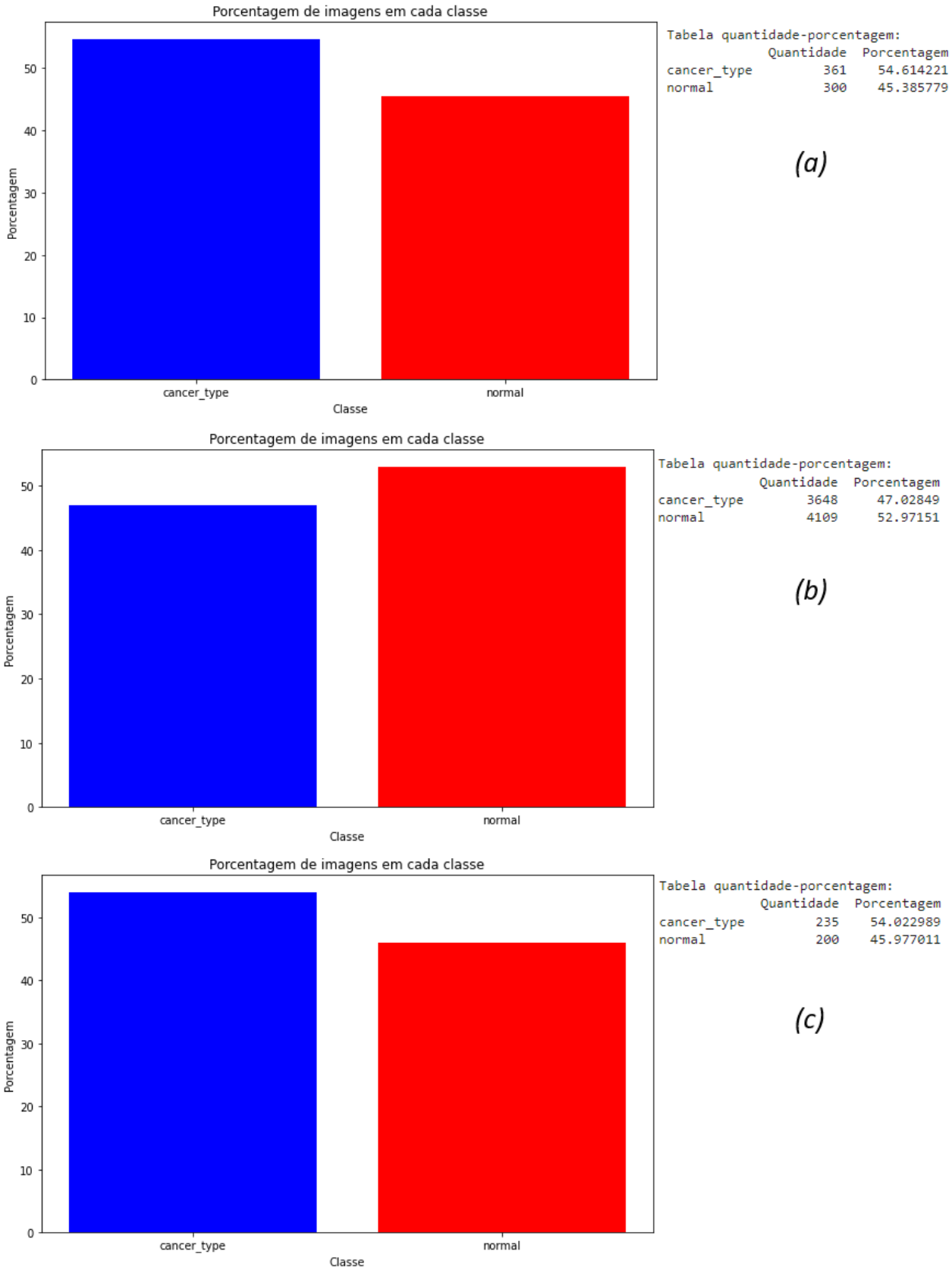

Figura 5.7. Base binária com *data augmentation* utilizada nos testes com a DenseNet-121. Essa configuração foi utilizada pois resultou em um bom balanceamento de dados. Os gráficos representam a base de teste (a), treinamento (b) e validação (c).

A rede utilizada foi a DenseNet-121 com seus pesos congelados e *batch size* de 32, seguida de uma operação de *flatten* e uma camada densa de classificação binária. Uitlizou-se o otimizador Adam com taxa de aprendizado de 0,001 e perda de entropia cruzada categórica. Por fim, foi usado *early stopping* com paciência de 25 épocas.

Os resultados do treinamento e validação são representados na Figura 5.8, que mostra que a rede teve novamente dificuldade na validação.

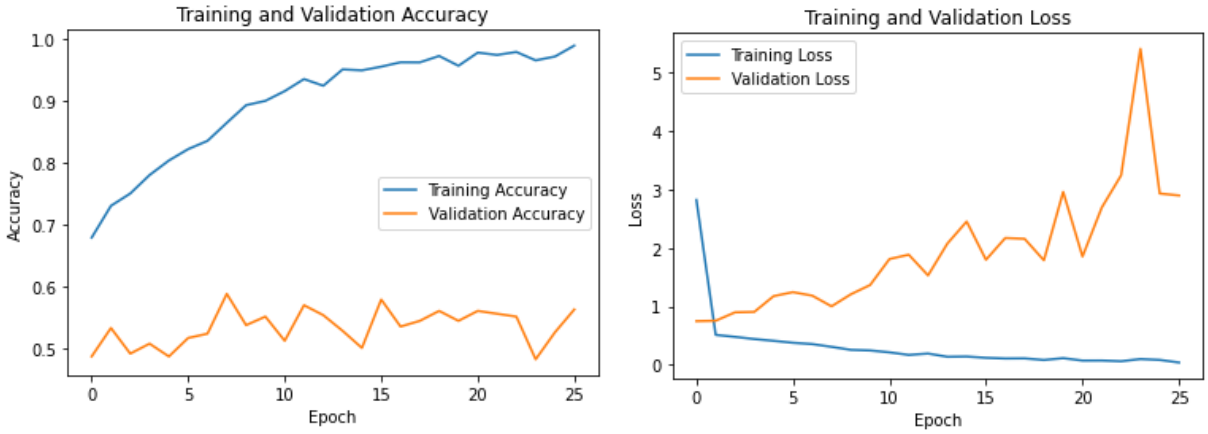

Figura 5.8. Teste e validação com DenseNet-121 com a base binária. Percebe-se dos gráficos que a rede chega próxima a um *overfitting*, mas o *early stopping* para o treinamento antes que a acurácia chegue em 100%.

Com essas configurações, a rede teve no teste uma acurácia de 54,01% e perda de 3,0925. A matriz de confusão gerada a partir das predições é representada na Figura 5.9.

Como os resultados das diversas arquiteturas testadas com diferentes redes mostraram-se insatisfatórios, foi realizada uma revisão do *dataset* e do *paper* para avaliar o que poderia ser feito para aprimorar tanto o treinamento como também a fase de testes das redes.

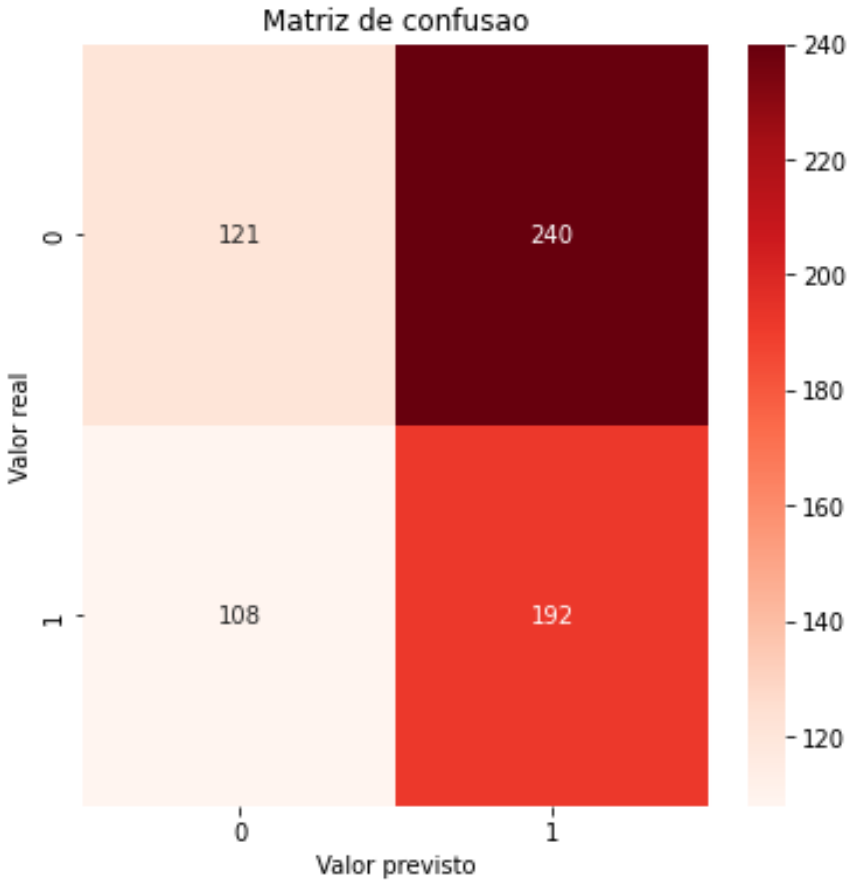

Figura 5.9. Matriz de confusão dos testes com DenseNet-121 com base binária. Percebe-se que a rede foi capaz de realizar classificações nas duas classes, mas ainda parece realizar chutes. Os 0s representam classificação de cancer, enquanto que os 1s representam o diagnóstico normal.

CAPÍTULO 6

TESTES FINAIS COM A BASE DE DADOS CORRIGIDA

A partir de uma releitura tanto do *paper* como também de uma análise mais aprofundada dos arquivos .csv disponibilizados pelo *dataset* descobriu-se dois fatos importantes acerca das imagens dos diagnósticos de câncer de mama.

O primeiro foi que a classe *actionable* representa um diagnóstico em que são necessários maiores análises da imagem a fim de definir qual seria a classificação apropriada, ou seja, não pode ser efetivamente considerada nem como câncer nem como normal. Por esse motivo, os próximos testes foram realizados sem essa classe (BUDA, 2021) (BUDA, 2024).

O segundo foi que as imagens de câncer têm apenas um de seus canais efetivamente diagnosticados por um radiologista com presença de tumor, visto que somente uma delas possui uma *bounding box* para segmentação associada. Devido a isso, tanto as imagens classficadas como *benign* e *cancer* foram convertidas novamente para .png a partir de seus arquivos .dcm da base de dados, mas agora extraindo o canal que o médico indicou efetivamente o câncer em vez do primeiro. As imagens com diagnóstico normal foram mantidas iguais (BUDA, 2021) (BUDA, 2024).

A nova base de dados é representadas na Figura 6.1. Como todos os testes anteriores que tiveram melhores resultados foram com classificações binárias, a mesma divisão foi feita aqui.

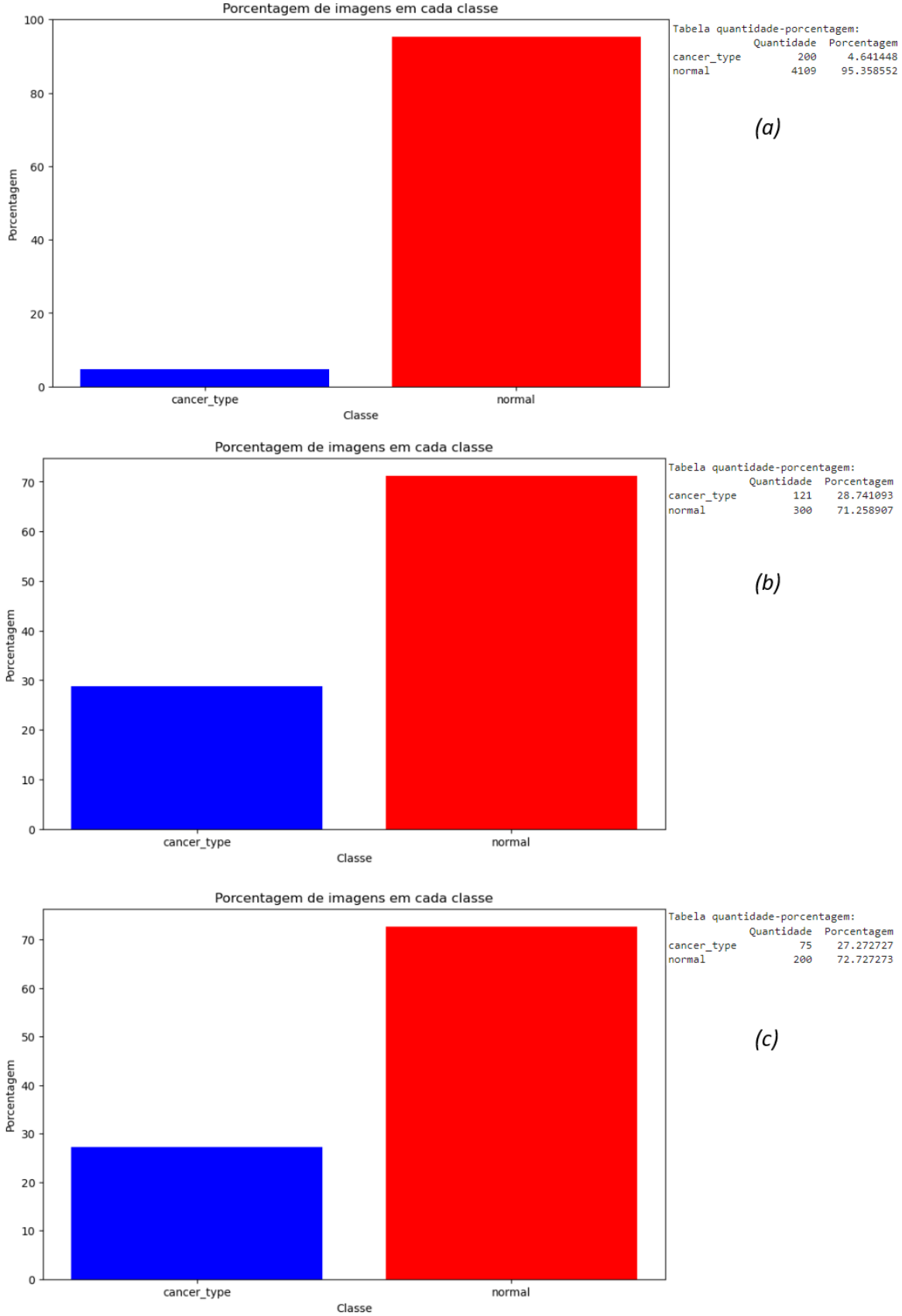

Figura 6.1. Base de dados binária dos testes finais. Nela foi retirada a classe *actionable* e os canais corretos das imagens classificadas como *benign* e *cancer* foram utilizadas. Os gráficos representam a base de treino (a), teste (b) e validação (c).

6.1 TESTES INICIAIS NA BASE DA DADOS CORRIGIDA

Os primeiros testes realizados com a nova base de dados foi com uso apenas do *transfer learning* das redes DenseNet-121, ResNet-50 e VGG-16 seguidas por uma operação de *flatten* e uma sigmoide para a classificação da imagem de entrada, a fim de avaliar como as redes sem mudanças iriam avaliar o *dataset* da Figura 6.1. Todos os pesos das redes foram mantidos congelados.

6.1.1 *Transfer learning* com ResNet-50

Para o aprendizado transferido com a rede ResNet-50, utilizou-se o otimizador Adam com taxa de aprendizado de 0,0005 e perda de entropia cruzada binária. Além disso, usou-se *early stopping* com uma paciência de 15 épocas e um *batch size* de 16. Por fim, utilizou-se a função de normalização própria do *tensorflow* para essa rede (tf.keras.applications.resnet.preprocess_input). Os resultados de treinamento e validação são representados na Figura 6.2.

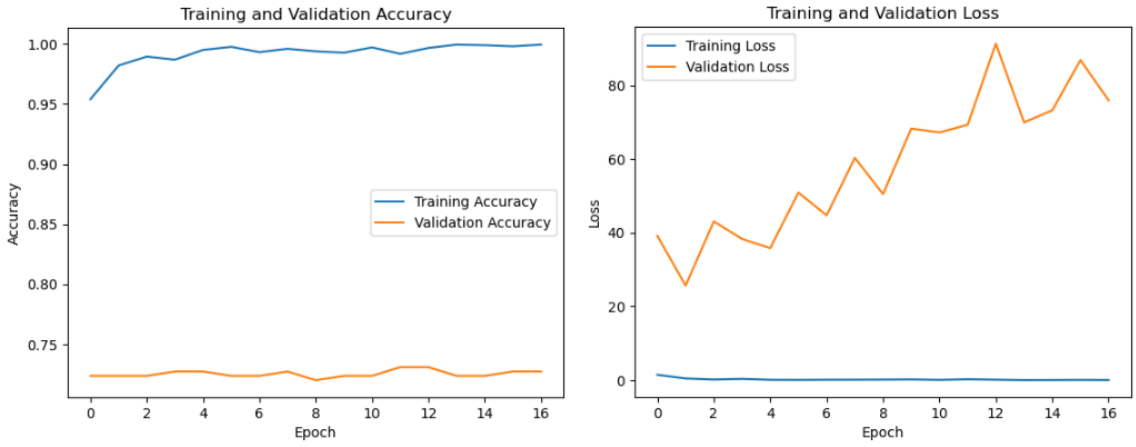

Figura 6.2. Resultados do treinamento e validação com a ResNet-50. Nota-se que assim como anteriormente, a rede aparenta resultar em *overfitting*, visto que não tem bom resultados no conjunto de validação.

Apesar dos resultados ruins em relação ao conjunto de validação, a rede teve uma boa acurácia de 91,69% e perda de 2,76 no conjunto de teste. Os resultados das predições são mostrados na Figura 6.3.

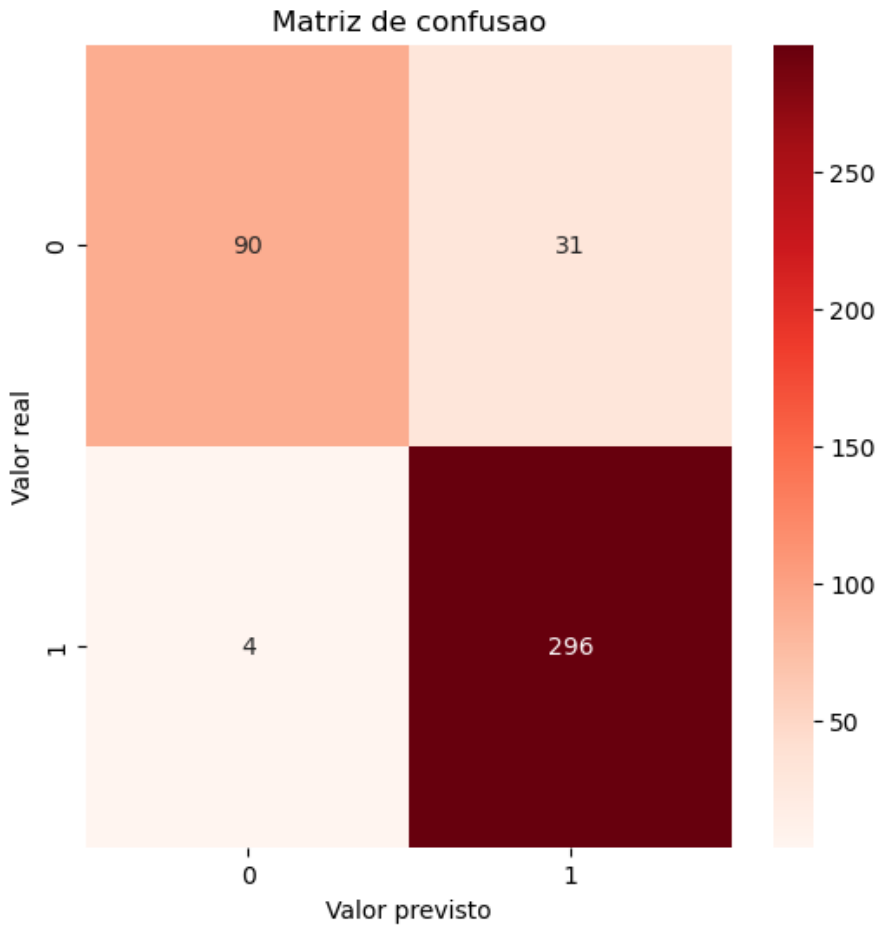

Figura 6.3. Matriz de confusão do conjunto de teste com a ResNet-50, em que os 0s representam os casos com câncer e os 1s com os casos normais. Apesar dos resultados ruins na validação, a rede teve bom desempenho no conjunto de teste.

Como a rede operou bem nos testes, foram calculadas também as outras métricas importantes para a avaliação da qualidade dela a partir dos valores da matriz de confusão da Figura 6.3. A diagonal principal representa os valores dos TPs e TNs, enquanto que a diagonal secundária representa os valores dos FNs e FPs. Os resultados das métricas são representados na Tabela 6.1.

Acurácia	**Especificidade**	**Sensibilidade**	**Precisão**	**F1-Score**
91,69%	98,67%	74,38%	95,74%	83,72%

Tabela 6.1. Tabela de métricas de desempenho da ResNet-50 calculadas a partir dos valores resultantes do conjunto de teste. Os resultados foram no geral bons, mas a sensibilidade está mais baixo que as outras.

Nota-se da Tabela 6.1 que a rede teve no geral bons resultados, mas o mais importante, a sensibilidade, não foi tão boa quanto as demais.

6.1.2 *Transfer learning* com a DenseNet-121

Para os testes com a DenseNet-121, as mesmas configurações que a ResNet-50 foram utilizadas, com exceção de uma taxa de aprendizado reduzida de 0,001 e a normalização utilizada anteriormente para essa rede, representada nas Equações 4.1 e 4.2. Os resultados do treinamento e validação com essa rede são mostrados na Figura 6.4.

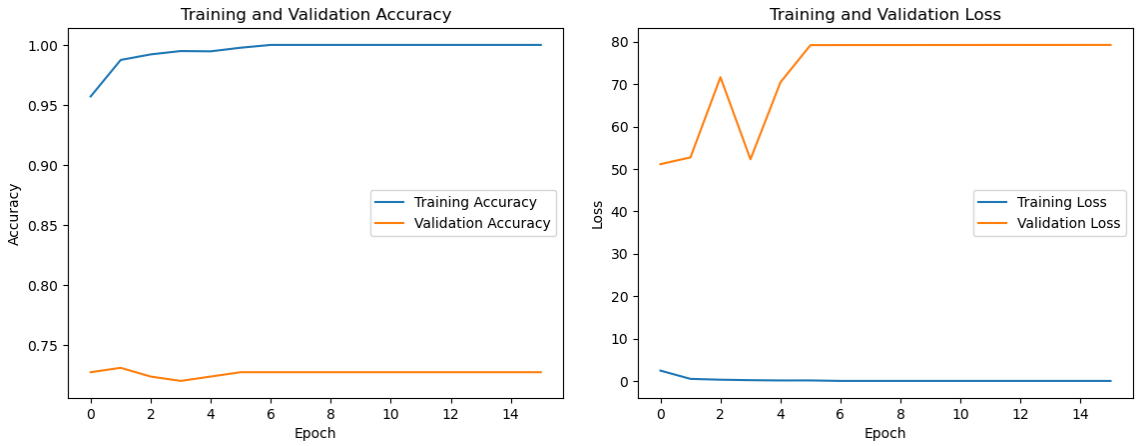

Figura 6.4. Resultados do treinamento e validação com a DenseNet-121. Novamente, a rede aparenta se aproximar de um *overfitting*, visto que vai bem no conjunto de treinamento, mas mal na validação.

Assim como anteriormente, a rede não teve bons resultados com o conjunto de validação, mas ficou cada vez melhor com o conjunto de treinamento. Nos testes ela teve predições que parecem ter aprendido um pouco as características das imagens com câncer, com acurácia de 85,99% e perda igual à 8,8. As predições realizadas podem ser visualizadas na Figura 6.5

Os resultados das métricas importantes para a avaliação da qualidade da rede foram calculada e são representadas na Tabela 6.2.

Acurácia	Especificidade	Sensibilidade	Precisão	F1-Score
85,99%	99%	53,72%	95,59%	68,78%

Tabela 6.2. Tabela de métricas de desempenho da DenseNet-121 calculadas a partir dos valores resultantes do conjunto de teste. Apesar de uma especificidade muito boa, sua sensibilidade está muito perto de um chute.

É perceptível dos valores da Tabela 6.2 como uma redução significativa da sensibilidade reduziu bastante o *F1-Score*.

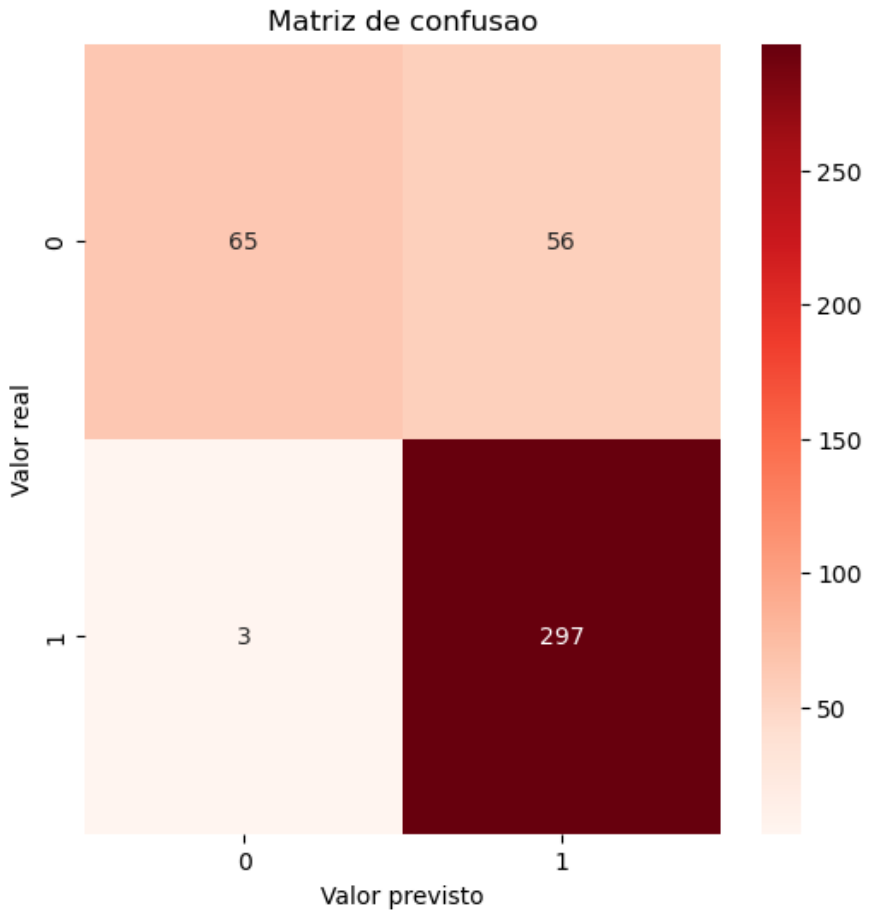

Figura 6.5. Matriz de confusão do conjunto de teste com a DenseNet-121. Nota-se das predições realizadas que a rede parece conseguir diferenciar algumas das imagens de câncer daquelas com diagnóstico normal, mas ainda não teve resultados muito bons.

6.1.3 *Transfer learning* com a VGG-16

Por fim, foram realizados testes com *transfer learning* com uso da rede VGG-16. As mesmas configurações utilizadas para a DenseNet-121 foram usadas aqui. Os resultados do treinamento e validação são apresentados na Figura 6.6.

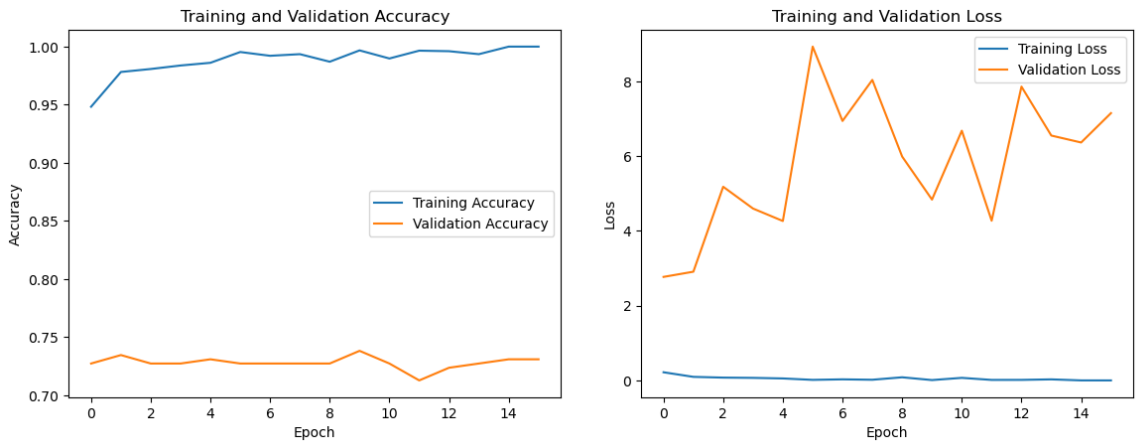

Figura 6.6. Resutado do treinamento e validação com a VGG-16. Nota-se menores perdas na validação para essa rede, quando comparado ao treinamento com a ResNet50 e a DenseNet-121.

É importante ressaltar que para a VGG-16, como pode ser visto na Figura 6.6, a perda da validação ficou muito menor quando comparada às duas redes anteriores, mas esmo assim, a acurácia da validação continuou baixa. Já nos testes, a rede teve uma boa acurácia de 85,04% e perda de 0,6077. As predições realizadas nesse conjunto são representadas na Figura 6.7

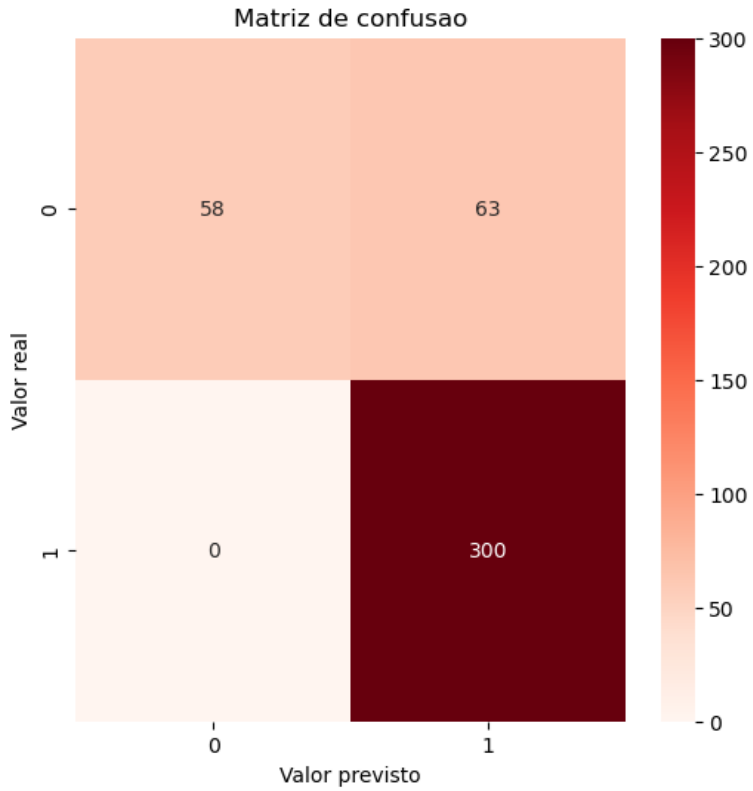

Figura 6.7. Matriz de confusão dos testes com a VGG-16. Assim como a ResNet-50, percebe-se que ela não teve uma boa sensibilidade.

A partir dos resultados dos testes, foram novamente calculadas as métricas importantes

para a avaliação da rede, que são representadas na Tabela 6.3

Acurácia	Especificidade	Sensibilidade	Precisão	F1-Score
85,04%	100%	47,93%	100%	64,8%

Tabela 6.3. Tabela de métricas de desempenho da VGG-16 calculadas a partir dos valores resultantes do conjunto de teste. Ela teve uma espeficidade perfeita, mas uma sensibilidade ruim.

Ao analisar-se os resultados obtidos na Tabela 6.3, percebe-se que a rede conseguiu uma especificidade e precisão perfeitas, mas uma sensibilidade muito ruim, que mais uma vez abaixou consideravelmente o *F1-Score*.

A partir dos resultados obtidos com as três redes, testou-se o uso de técnicas de regularização com a base de dados corrigida, a fim de verificiar a possibilidade de melhorias no desempenho das redes, principalmente na sensibilidade, que representa a métrica mais importante para o propósito do trabalho.

6.2 TESTES COM PERDA PONDERADA NA BASE DE DADOS CORRIGIDA

Foram realizados diferentes testes com diferentes formas de regularização e diferentes configurações de redes, mas os que trouxeram os melhores resultados foi a partir do uso da técnica de regularização da função de perda ponderada junto com o *transfer learning* das redes seguidas por uma operação de *flatten* e sigmoide. Para todas elas, um *batch size* de 16 foi utilizado junto com um *early stopping* com paciência de 20 épocas e perda de entropia cruzada binária.

Os pesos da técnica de perda ponderada foram configurados a partir da quantidade de dados em cada classe presente no conjunto de treinamento, representado na Figura 6.1. A partir disso, eles são calculados a partir da Equação 3.42 como:

$$\omega_0 = \frac{1}{200} \times \frac{4309}{2} = 10{,}77 \tag{6.1}$$

$$\omega_1 = \frac{1}{4109} \times \frac{4309}{2} = 0{,}52 \tag{6.2}$$

em que ω_0 representa os pesos para a classe minoritária *cancer type* e ω_1 representa os pesos para a classe majoritária normal.

6.2.1 *Transfer learning* com a ResNet-50

Para a ResNet-50, foi utilizado o otimizador Adam com uma taxa de aprendizagem de 0,0005 e foi normalizada a com a função de otimização fornecida pelo *tensorflow*. Os resultados de treinamento e validação são apresentados na Figura 6.8

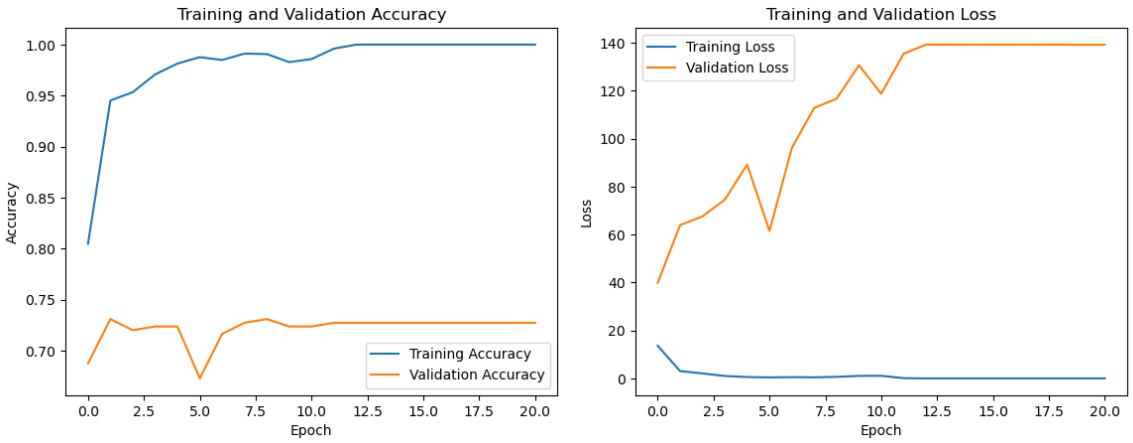

Figura 6.8. Resultados do treinamento e validação com a ResNet-50 na base de dados corrigida. Nota-se que a rede aparenta resultar em *overfitting*, visto que não tem bom resultados no conjunto de validação.

A partir dos gráficos da Figura 6.8, é notável que mais uma vez a rede não se sai bem no conjunto de validação, mas chega em um *overfitting* no treinamento. Apesar disso, ela teve bons resultados no teste, com acurácia de 90,02% e perda de 5,35. As predições realizadas são mostradas na Figura 6.9.

Nota-se da matriz de confusão da Figura 6.9 que no conjunto de teste a rede foi capaz de prever as classes das imagens de forma eficiente quando comparada aos testes anteriores. As métricas de qualidade calculadas são representadas na Tabela 6.4.

Acurácia	Especificidade	Sensibilidade	Precisão	F1-Score
90,02%	93%	82,64%	82,64%	82,54%

Tabela 6.4. Tabela de métricas de desempenho da ResNet-50 na base corrigida. É notável como tanto a sensibilidade como também o *F1-Score* tiveram resultados melhores, apesar da redução da especificidade.

Os valores calculados das métricas de qualidade da rede mostram que houve uma melhoria considerável na capacidade da rede de detectar as imagens com presença de tumores, apesar de uma pequena piora em sua especificidade.

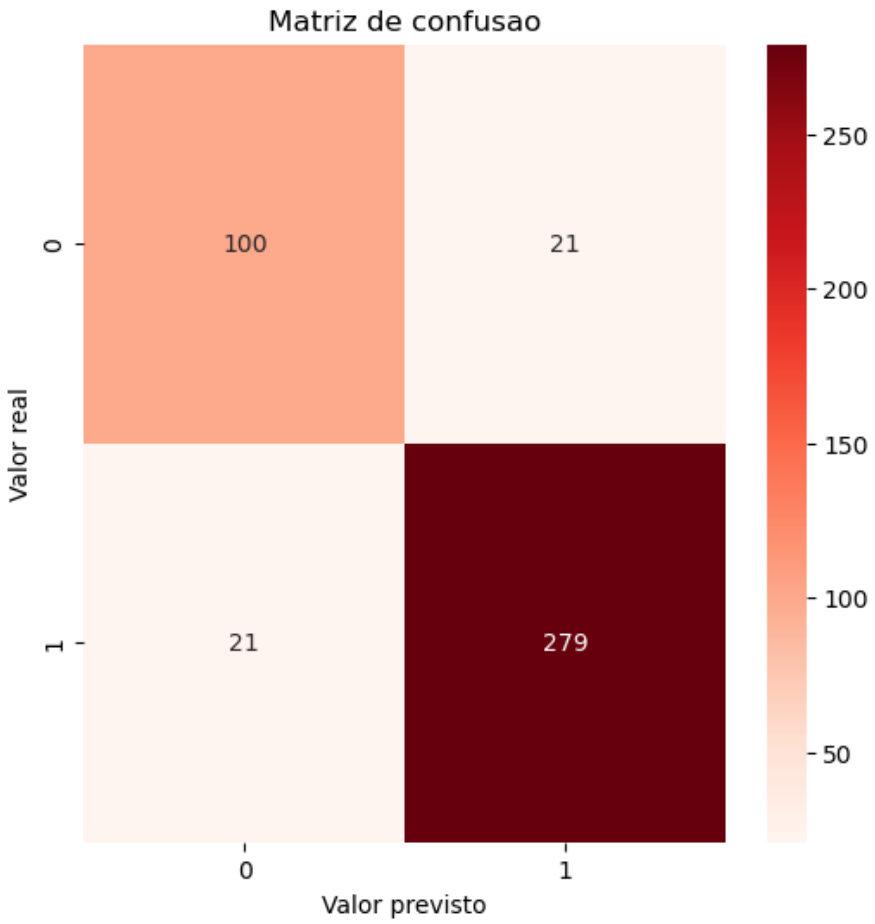

Figura 6.9. Matriz de confusão da ResNet-50 na base de dados corrigida. Percebe-se que no teste, a rede teve bons resultados, visto que conseguiu prever corretamente uma boa quantidade das imagens.

6.2.2 *Transfer learning* com a DenseNet-121

As mesmas configurações para a rede utilizadas na subseção 6.1.2 foram utilizadas aqui. Após o treinamento e validação, os gráficos representados na Figura 6.10 mostram sua acurácia e perda nesses dois conjuntos.

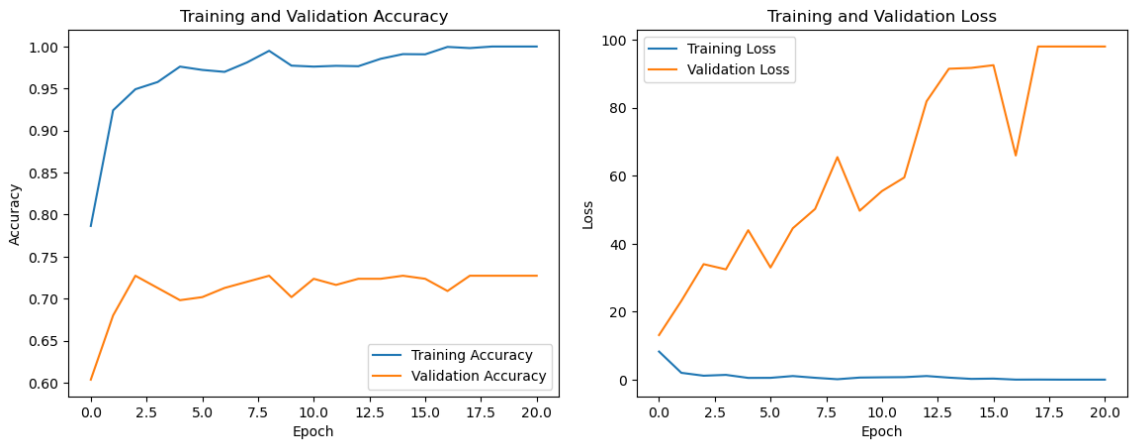

Figura 6.10. Treinamento e validação da DenseNet-121 na base de dados corrigida. Mais uma vez, a rede não teve bons valores de perda para o conjunto de validação.

Mais uma vez, é perceptível como a rede falhou em conseguir compreender o conjunto de validação. Contudo, ela se saiu um pouco melhor no conjunto de teste, com uma acurácia de 75,77% e perda de 6,45. As predições realizadas são representadas na matriz de confusão da Figura 6.11

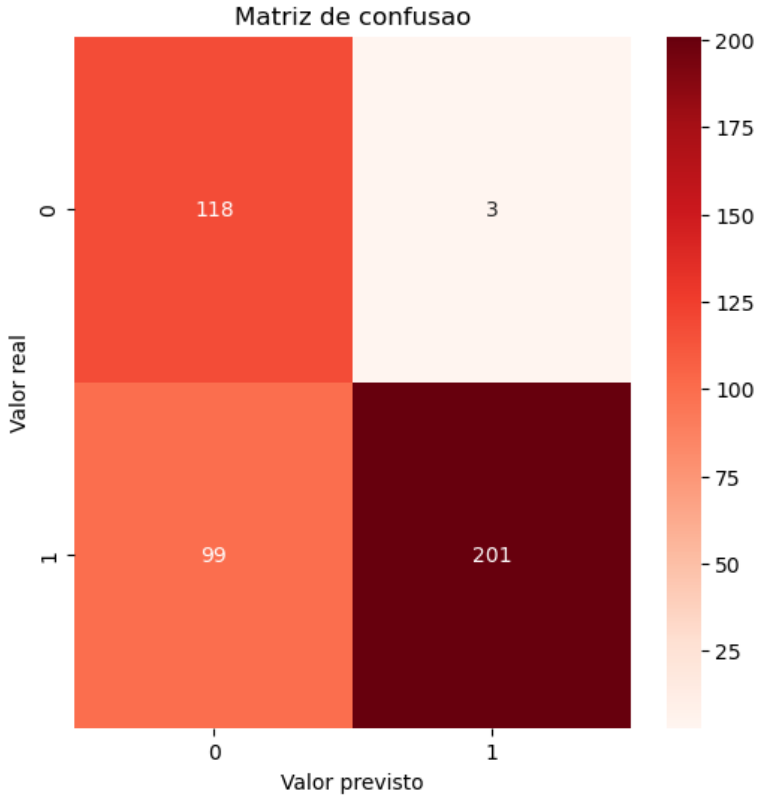

Figura 6.11. Matriz de confusão com a base corrigida da DenseNet-121. Nota-se das predições realizadas que a rede conseguiu classificar muito bem as imagens de câncer, mas não as classificadas como normais.

Nota-se dos resultados das predições como a rede conseguiu classificar muito bem as imagens

de câncer, mas piorou na classificação das imagens normais. As métricas calculadas estão representadas na Tabela 6.5.

Acurácia	Especificidade	Sensibilidade	Precisão	F1-Score
75,77%	67%	97,52%	54,38%	69,82%

Tabela 6.5. Tabela de métricas de desempenho da DenseNet-121 na base corrigida. Percebe-se como houve um desempenho alto em relação à sensibilidade, mas a queda na especificidade fez com que o *F1-Score* ficasse baixo.

A partir dos resultados da Tabela 6.5, é perceptível que a rede teve um desempenho muito bom em relação à sensibilidade, mas teve um redução grande em sua especificidade, precisão e *F1-Score*.

6.2.3 *Transfer learning* com a VGG-16

As mesmas configurações para a rede utilizadas na subseção 6.1.3 foram utilizadas aqui. Após o treinamento e validação, os gráficos representados na Figura 6.10 mostram sua acurácia e perda nesses dois conjuntos.

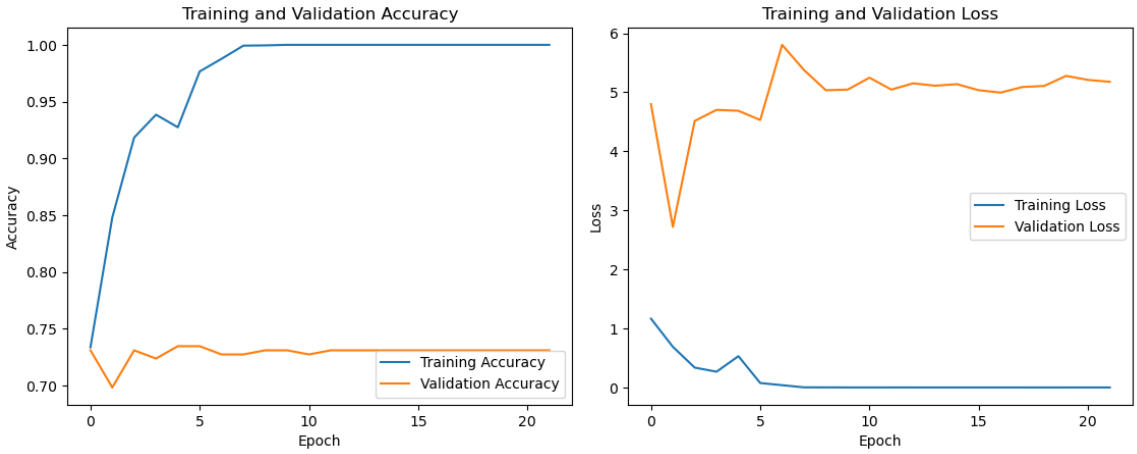

Figura 6.12. Treinamento e validação da VGG-16 na base de dados corrigida. Novamente, a rede não teve bons valores de perda para o conjunto de validação, mas foi bem no treinamento. Mais uma vez, a VGG-16 se mostra com uma perda significantivamente menor do que as outras redes na validação.

Apesar da ineficácia da rede durante o cojunto de validação, ela se mostrou boa ao realizar as predições com o conjunto de teste, com uma acurácia de 90,26% e perda de 0,355. Além disso, quando comparada às outras duas redes, sua perda é signicativamente menor. As predições realizadas durante o teste podem ser visualizadas na Figura 6.13

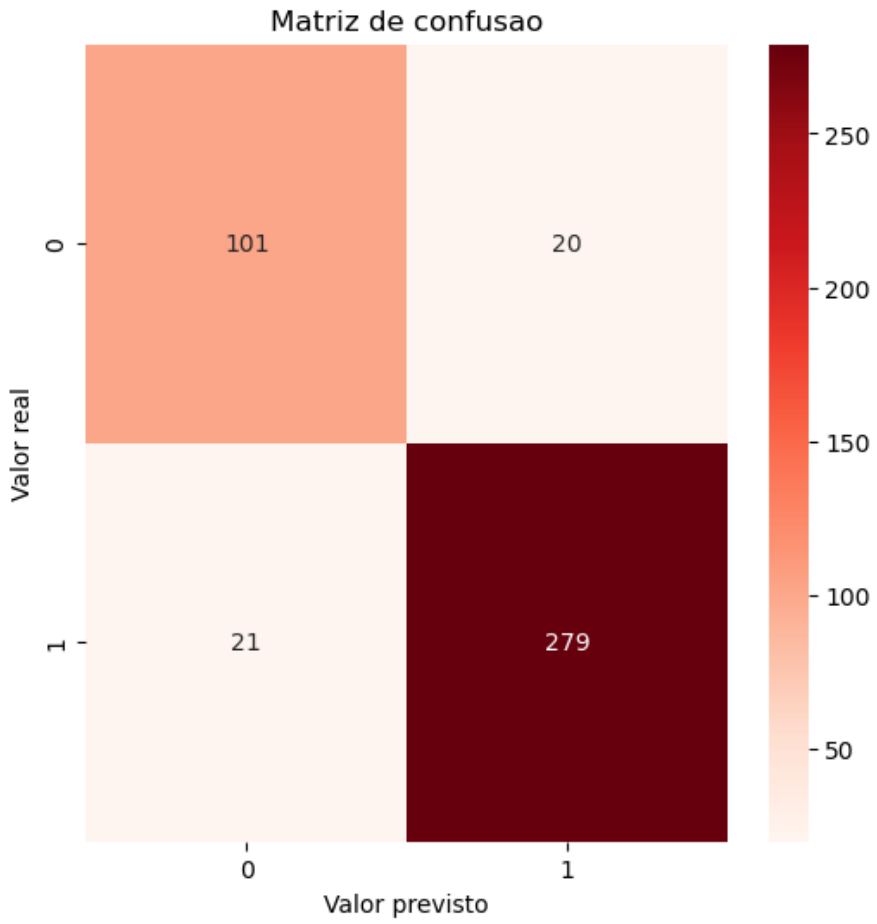

Figura 6.13. Matriz de confusão com a base corrigida da VGG-16. É notável que, no geral, a rede conseguiu prever corretamente as classes das imagens.

A partir da matriz de confusão da Figura 6.13, percebe-se que a rede teve bons resultados nas predições das classes das imagens da base de dados. As métricas calulcadas para essa rede no conjunto de teste estão representadas na Tabela 6.6.

Acurácia	Especificidade	Sensibilidade	Precisão	F1-Score
90,26%	93%	83,47%	82,79%	83,13%

Tabela 6.6. Tabela de métricas de desempenho da VGG-16 na base corrigida. No geral, as métricas de qualidade da rede tiveram bons valores.

Pode-se perceber dos valores da Tabela 6.6 e da perda de treinamento que a rede teve um bom desempenho na classificação das imagens da base de dados corrigia.

6.2.4 Comparação dos resultados com a base de dados corrigida

Por fim, a comparação dos resultados das redes desta seção são representadas para uma comparação fácil na Tabela 6.7.

Modelo	Weighted Loss	Acurácia	Especificidade	Sensibilidade	Precisão	F1-score
DenseNet-121	Não	86%	99%	54%	96%	69%
DenseNet-121	Sim	76%	67%	98%	54%	70%
ResNet-50	Não	92%	99%	74%	96%	84%
ResNet-50	Sim	90%	93%	83%	83%	83%
VGG-16	Não	85%	100%	48%	100%	65%
VGG-16	Sim	90%	93%	83%	83%	83%

Tabela 6.7. Comparação de desempenho das redes neurais com a base de dados corrigida, com e sem o uso de perda ponderada.

A partir dos resultados obtidos, é notável como o uso de técnicas de regularização afetam os resultados do teste das redes, principalmente com melhorias nos valores da sensbilidade, no geral em detrimento da especificidade. Destaca-se principalmente a VGG-16 com uso da perda ponderada, que teve uma grande melhoria em sua capacidade de classificar corretamente as imagens com tumores.

CONCLUSÕES E TRABALHOS FUTUROS

A partir dos resultados obtidos neste trabalho, nota-se como o uso de CNNs, principalmente quando desenvolvidas junto com técnicas de regularização, foi capaz de classificar com certa eficácia as imagens das mamografias. A VGG-16 com perda ponderada se destacou, visto que teve um valor de *loss* baixa (quando comparada aos outros modelos) durante o teste e bons resultados gerais em suas predições quando comparados com as outras configurações das redes, como mostra a Tabela 6.7.

Além disso, também foi notável no decorrer do trabalho as dificuldades relacionadas a encontrar configurações de redes neurais apropriadas para realizar as tarefas de classificação de imagens, visto que diversas arquiteturas e ajustes finos diferentes foram testados até encontrar um padrão que conseguisse diferenciar as características presentes nas mamografias.

A partir dos resultados com a VGG-16, percebeu-se também como não necessariamente uma rede mais complexa ou mais profunda será a melhor para toda tarefa, visto que, dentre as três redes utilizadas neste trabalho, ela possui a arquitetura mais simples, que pode ser observada na Figura 3.19.

Outrossim, foi possível também perceber como é de grande importância não só realizar apropriadamente o pré-processamento das imagens das bases de dados, mas também como a organização correta das classes das mamografias possibilitou uma melhoria significativa no desempenho das três redes utilizadas no trabalho.

No entanto, para que o uso dessas CNNs seja considerado viável em casos clínicos, é necessário melhorar ainda mais os resultados das classificações, especialmente em termos de sensibilidade. A capacidade da rede em identificar corretamente os casos de câncer em situações reais é crucial para a qualidade de vida das pacientes.

Trabalhos futuros poderiam se concentrar na melhoria das arquiteturas de redes neurais e na integração de novas técnicas de pré-processamento de imagens, visto que não foram aproveitados

100% dos dados disponíveis do *dataset*, já que neste projeto apenas um dos canais de cada uma das imagens foi utilizado no treinamento, validação e teste das CNNs.

Além disso, seria de grande importância para próximas pesquisas avaliar e corrigir os possíveis motivos por trás da grande discrepância do desempenho das redes nos três conjuntos, visto que todas elas tiveram resultados ruins na validação, independente da configuração utilizada.

Por fim, também seria recomendável para trabalhos futuros obter mais opiniões de radiologistas acerca das imagens classificadas como *actionable* na base de dados, visto que poderiam contribuir para um melhor desepenho no treinamento da rede caso fossem classificadas dentro dos outros três diagnósticos, já que isso aumentaria a quantidade de imagens disponíveis.

REFERÊNCIAS BIBLIOGRÁFICAS

AGUILLAR, V. L. N. *Breast Tomosynthesis - A better mamography*. 2018. Disponível em: <https://pesquisa.bvsalud.org/portal/resource/pt/biblio-915932>. Citado na página 6.

BUDA, M. *A Data Set and Deep Learning Algorithm for the Detection of Masses and Architectural Distortions in Digital Breast Tomosynthesis Images*. 2021. Disponível em: <https://jamanetwork.com/journals/jamanetworkopen/fullarticle/2783046>. Citado 5 vezes nas páginas 14, 59, 62, 63, and 71.

BUDA, M. M. *Breast Cancer Screening – Digital Breast Tomosynthesis (BCS-DBT) (Version 5) [dataset]. The Cancer Imaging Archive*. 2024. Disponível em: <https://www.cancerimagingarchive.net/collection/breast-cancer-screening-dbt/#citations>. Citado 5 vezes nas páginas 4, 9, 12, 13, and 71.

COLANGELO, M. *How AI Can Help Address The Global Shortage of Radiologists*. 2022. Disponível em: <https://www.linkedin.com/pulse/how-ai-can-help-address-global-shortage-radiologists-colangelo/>. Citado 2 vezes nas páginas 3 and 4.

D, D. *Zero-Padding in Convolutional Neural Networks*. 2021. Disponível em: <https://medium.com/@draj0718/zero-padding-in-convolutional-neural-networks-bf1410438e99>. Citado na página 18.

DOC, A. *The Radiologist Shortage and the Potential of AI*. 2022. Disponível em: <https://www.aidoc.com/learn/blog/is-radiologist-shortage-real/>. Citado na página 3.

G, R. *Everything you need to know about VGG16*. 2021. Disponível em: <https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918>. Citado na página 42.

GENCAY, R. *Early stopping based on cross-validation*. 2021. Disponível em: <https://www.researchgate.net/figure/Early-stopping-based-on-cross-validation_fig1_3302948>. Citado na página 35.

GUDIMALLA, V. *Concept of Gradient Descent in Machine Learning*. 2021. Disponível em: <https://varshithagudimalla.medium.com/concept-of-gradient-descent-algorithm-in-machine-learning-44f587ac16ac>. Citado 2 vezes nas páginas 24 and 25.

HE, K. *Deep Residual Learning for Image Recognition*. 2015. Disponível em: <https://arxiv.org/abs/1512.03385>. Citado 3 vezes nas páginas 4, 40, and 41.

HELVIE, M. M. A. *Digital Mammography Imaging: Breast Tomosynthesis and Advanced Applications*. 2011. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3118307/#:~:text=In%20digital%20breast%20tomosynthesis%2C%20the,used%20during%20conventional%20digital%20mammography>. Citado 5 vezes nas páginas 6, 7, 8, 9, and 10.

HOSNI, Y. *Maximizing the Impact of Data Augmentation: Effective Techniques and Best Practices*. 2023. Disponível em: <https://pub.towardsai.net/maximizing-the-impact-of-data-augmentation-effective-techniques-and-best-practices-c4cad9cd16e4>. Citado na página 36.

HUANG, G. *Densely Connected Convolutional Networks*. 2016. Disponível em: <https://arxiv.org/abs/1608.06993>. Citado 3 vezes nas páginas 4, 38, and 39.

KUMAR, B. *Convolutional Neural Networks: A Brief History of their Evolution*. 2021. Disponível em: <https://medium.com/appyhigh-technology-blog/convolutional-neural-networks-a-brief-history-of-their-evolution-ee3405568597>. Citado na página 15.

LANES, F. *Entenda porque há déficit de radiologistas em algumas regiões do Brasil*. 2022. Disponível em: <https://www.telelaudo.com.br/blog/superar-deficit-radiologistas-brasil/>. Citado na página 3.

LECUN, Y. *Gradient-Based Learning Applied to Document Recognition*. 1998. Disponível em: <https://www.researchgate.net/publication/2985446_Gradient-Based_Learning_Applied_to_Document_Recognition>. Citado 3 vezes nas páginas 24, 25, and 27.

LECUN, Y. *Efficient Backprop*. 2000. Disponível em: <https://www.researchgate.net/publication/2811922_Efficient_BackProp>. Citado 2 vezes nas páginas 23 and 26.

NG, A. *Deep Learning Specialization*. 2024. Disponível em: <https://www.coursera.org/learn/convolutional-neural-networks?specialization=deep-learning>. Citado 26 vezes nas páginas 3, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 40, 41, and 42.

O'SHEA, K.; NASH, R. *An Introduction to Convolutional Neural Networks*. 2015. Disponível em: <https://arxiv.org/abs/1511.08458>. Citado 2 vezes nas páginas 16 and 22.

PATEL, A. *Benign vs Malignant Tumors*. 2020. Disponível em: <https://jamanetwork.com/journals/jamaoncology/fullarticle/2768634#:~:text=Benign%20tumors%20are%20those%20that,tumors%20are%20not%20usually%20problematic.> Citado na página 13.

PATEL, A. *Implementing DenseNet-121 in PyTorch: A Step-by-Step Guide*. 2023. Disponível em: <https://medium.com/deepkapha-notes/implementing-densenet-121-in-pytorch-a-step-by-step-guide-c0c2625c2a60>. Citado na página 40.

REYNOLDS, A. H. *Convolutional Neural Networks (CNNs)*. 2019. Disponível em: <https://anhreynolds.com/blogs/cnn.html>. Citado 3 vezes nas páginas 17, 18, and 19.

SAVYAKHOSLA. *Introduction to Pooling Layers*. 2023. Disponível em: <https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/>. Citado 2 vezes nas páginas 20 and 21.

SIMONYAN, K. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. Disponível em: <https://arxiv.org/abs/1409.1556>. Citado 2 vezes nas páginas 4 and 42.

VARMA, D. R. *Managing DICOM images: Tips and tricks for the radiologist*. 2012. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3354356/#:~:text=A%20DICOM%20file%20consists%20of,demographics%2C%20study%20parameters%2C%20etc>. Citado 4 vezes nas páginas 6, 10, 11, and 12.

XU, Y. *Global trends and forecasts of breast cancer incidence and deaths*. 2023. Disponível em: <https://www.nature.com/articles/s41597-023-02253-5>. Citado 2 vezes nas páginas 3 and 4.